VŠB – Technická univerzita Ostrava
Fakulta strojní
Katedra robototechniky

Robotizované výukové pracoviště s robotem ABB IRB 360

Robotized Educational Cell with Robot ABB IRB 360

Student: Bc. Jakub Mžik
Vedoucí diplomové práce: Ing. Václav Krys, Ph.D.

Ostrava 2013

VŠB - Technická univerzita Ostrava
Fakulta strojní
Katedra robototechniky

Zadání diplomové práce

Student: **Bc. Jakub Mžik**

Studijní program: N2301 Strojní inženýrství

Studijní obor: 2301T013 Robotika

Téma: Robotizované výukové pracoviště s robotem ABB IRB 360
Robotized Educational Cell with Robot ABB IRB 360

Zásady pro vypracování:

1. Popište a uveďte základní technické parametry komponent výukového robotizovaného pracoviště s IRB 360.

2. Navrhněte a realizujte alespoň 3 výukové úlohy pro toto pracoviště včetně periferií.

3. Realizované úlohy detailně popište a v elektronické formě doložte kompletní dokumentaci k programům.

4. Práci též doložte v elektronické podobě ve formátu MS WORD.

Seznam doporučené odborné literatury:

ČSN 01 6910 *Úprava písemností psaných strojem nebo zpracovaných textovými editory.* Praha: Český normalizační institut, srpen 2007. 48 s.

ČSN ISO 690 *Informace a dokumentace - Pravidla pro bibliografické odkazy a citace informačních zdrojů.* Praha: Český normalizační institut, 2010.

Manuály ABB dodané s robotizovaným pracovištěm.

Formální náležitosti a rozsah diplomové práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí diplomové práce: **Ing. Václav Krys, Ph.D.**

Datum zadání: 31.10.2012

Datum odevzdání: 20.05.2013

VYSOKÁ ŠKOLA BÁŇSKÁ-TECHNICKÁ UNIVERZITA Fakulta strojní 3 OSTRAVA

prof. Dr.Ing. Petr Novák
vedoucí katedry

doc. Ing. Ivo Hlavatý, Ph.D.
děkan fakulty

Místopřísežné prohlášení studenta

Prohlašuji, že jsem celou diplomovou práci včetně příloh vypracoval samostatně pod vedením vedoucího diplomové práce a uvedl jsem všechny použité podklady a literaturu.

V Ostravě 17.5.2013 .....

.....

Bc. Jakub Mžik

Prohlašuji, že:

• jsem byl seznámen s tím, že na moji diplomovou práci se plně vztahuje zákon č. 121/2000 Sb., autorský zákon, zejména § 35 – užití díla v rámci občanských a náboženských obřadů, v rámci školních představení a užití díla školního a § 60 – školní dílo.

• beru na vědomí, že Vysoká škola báňská – Technická univerzita Ostrava (dále jen „VŠB-TUO“) má právo nevýdělečně ke své vnitřní potřebě diplomovou práci užít (§ 35 odst. 3).

• souhlasím s tím, že diplomová práce bude v elektronické podobě uložena v Ústřední knihovně VŠB-TUO k nahlédnutí a jeden výtisk bude uložen u vedoucího diplomové práce. Souhlasím s tím, že údaje o kvalifikační práci budou zveřejněny v informačním systému VŠB-TUO.

• bylo sjednáno, že s VŠB-TUO, v případě zájmu z její strany, uzavřu licenční smlouvu s oprávněním užít dílo v rozsahu § 12 odst. 4 autorského zákona.

• bylo sjednáno, že užít své dílo – diplomovou práci nebo poskytnout licenci k jejímu využití mohu jen se souhlasem VŠB-TUO, která je oprávněna v takovém případě ode mne požadovat přiměřený příspěvek na úhradu nákladů, které byly VŠB-TUO na vytvoření díla vynaloženy (až do jejich skutečné výše).

• beru na vědomí, že odevzdáním své práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb., o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších předpisů, bez ohledu na výsledek její obhajoby.

V Ostravě 17.5.2013 .....

.....
Bc. Jakub Mžik

Bc. Jakub Mžik
U Stanoviště 1122
735 53, Dolní Lutyně

Poděkování

Tato práce byla vypracována s podporou projektu Příležitost pro mladé výzkumníky, reg. č. CZ.1.07/2.3.00/30.0016, podpořeného Operačním programem Vzdělávání pro konkurenceschopnost a spolufinancovaného Evropským sociálním fondem a státním rozpočtem České republiky.

Děkuji především panu Ing. Václavu Krysovi, Ph.D. za poskytnuté konzultace při přípravě mé diplomové práce. Dále děkuji kolegům Bc. Tomáši Chamradovi a Bc. Martinu Soukupovi za spolupráci při realizaci.

Anotace diplomové práce

MŽIK, J. *Robotizované výukové pracoviště s robotem ABB IRB 360: diplomová práce.* Ostrava: VŠB –Technická univerzita Ostrava, Fakulta strojní, Katedra robototechniky, 2013, 74s. Vedoucí práce: Krys, V.

Diplomová práce se zabývá pokročilými metodami programování robotu ABB IRB 360 FlexPicker v součinnosti se softwarem ABB PickMaster 3. V úvodu jsou popsány základní prvky robotizovaného pracoviště na Centru robotiky Katedry robototechniky VŠB-TU Ostrava. Dále je popsána konstrukce a zapojení dodatečně instalovaných periferií. Hlavní částí je pak popis průběhu standardního programu a metody vstupu do něj za účelem uživatelských modifikací. Tyto metody jsou demonstrovány na dvou úlohách napodobujících použití v praxi. Jako třetí úloha je realizováno vyskládání znaků do matrice s možností uživatelského vstupu pomocí FlexPendantu nebo mobilního telefonu a rozhraní Bluetooth. Všechny úlohy jsou detailně popsány a v elektronické podobě je přiložena jejich kompletní dokumentace.

Annotation of diploma thesis

MŽIK, J. *Robotized Educational Cell with Robot ABB IRB 360: diploma thesis.* Ostrava: VŠB – Technical university of Ostrava, Faculty of Mechanical Engineering, Department of Robotics, 2013, 74p. Thesis head: Krys, V.

This thesis deals with advanced methods of programming of the ABB IRB 360 FlexPicker robot in cooperation with the ABB PickMaster 3 software. The introduction describes the basic components of the robotized workplace at the Robotics Centre of the Department of Robotics, VSB-TU Ostrava. It also describes the design and connection of the additionally installed peripherals. The main part is the description of the standard program and methods of its user modifications. These methods are demonstrated on two tasks imitating usage in praxis. As the third task, inscribing of characters in a matrix is realized with user input via FlexPendant or mobile phone and Bluetooth. All tasks are described in detail and a complete documentation in electronic form is included.

Obsah

1 Úvod 10

2 Cíle diplomové práce 11

3 Pracoviště s ABB IRB 360 FlexPicker 12

3.1 Pracoviště na centru robotiky 12

3.2 ABB IRB 360 FlexPicker 16

3.3 PickMaster 3 18

4 Instalované periferie 19

4.1 Světelný box a odkládací stoly 19

4.2 Panel ovládání pracovních prostorů 21

4.2.1 Funkce a vnitřní elektrické zapojení 22

4.2.2 Připojení k řadiči a konfigurace 24

4.3 Bluetooth modul 27

4.3.1 Elektrické zapojení 27

4.3.2 Konfigurace modulu 30

4.3.3 Sériová komunikace s řadičem robotu 30

5 Standardní program a vstup do něj 35

5.1 Popis jednotlivých rutin 37

5.2 Item Source Data 42

5.3 Funkce TriggL a její využití při časování pohybů 43

5.3.1 Obecně o funkci TriggL 43

5.3.2 Časování při pohybech robotu 46

5.4 Způsoby editace RAPID kódu 48

5.4.1 Editace přímo v prostředí PickMaster 3 a textovém editoru 48

5.4.2 Editace v dočasných modulech v RobotStudiu 49

5.4.3 Použití systémových modulů 51

6 Úlohy k realizaci 53

6.1 Skládání objektů do komínků 53

6.2 Třídění žetonů podle barev 56

6.3 Skládání znaků do matrice 57

6.3.1 Moduly úlohy skládání znaků do matrice 58
6.3.2 Matrice a princip práce s nimi 59
6.3.3 Znaky a jejich definice 61
6.3.4 Algoritmus skládání znaku 62
6.3.5 HMI na FlexPendantu 65
6.3.6 Simulace v RobotStudiu 66
7 Závěr 70
8 Zdroje 71
9 Seznam příloh 73

Zkratky a pojmy

BT	Bluetooth
Conveyor Tracking	Systém sledování dopravníku
DI	Digital Input - digitální vstup (signálu)
DO	Digital Output - digitální výstup (signálu)
HMI	Human-Machine Interface - rozhraní člověk-stroj (uživatelské rozhraní)
Item	Obecný název pro rozpoznaný objekt manipulace
I/O	Input / Output (vstupní / výstupní)
Jointtarget	Datový typ - cílové kloubové proměnné do kterých se polohuje robot
LTM	Levé tlačítko myši
OM	Objekt manipulace
PC	Stolní počítač
PTM	Pravé tlačítko myši
RAPID	Programovací jazyk robotů ABB
Robtarget	Datový typ - cílový bod do kterého se polohuje koncový bod robotu (TCP)
Softbox	Světelný box- používá se k rovnoměrnému rozptylu světla na osvětlovaný objekt
TCP	Tool Center Point - středový bod nástroje (většinou jeho konec, např. hrot drátu u svařovací pistole, střed přísavky)
LCR	Laboratoř na centru robotiky
WA	Work Area - pracovní prostor

1 Úvod

Robotizovaná pracoviště pro rychlé odebírání výrobků a jejich uspořádávání (*Picking and Placing*) zaznamenávají v současnosti, zejména v potravinářství, velký nárůst počtu aplikací. Za tímto samozřejmě stojí zejména trend robotizace (či obecně automatizace) výrobních procesů s cílem snižování výrobních nákladů. Vedle toho je tento fakt zřejmě dán
i intenzitou vývoje na poli rozpoznávání obrazu. Tyto aplikace totiž vyžadují poměrně velké výpočetní výkony, které v minulosti nebyl schopen hardware za rozumnou cenu poskytnout.

Pro malé objekty s nízkými hmotnostmi (cca do 2 kg) se jako vhodné ukázaly paralelní struktury, takzvané delta roboty. Jejich struktura jim dovoluje dosahovat velmi vysokých rychlostí a zrychlení i při poměrně jednoduché konstrukci. Jedním z nich je i ABB IRB 360 FlexPicker instalovaný v Laboratoři na centru robotiky Katedry robototechniky.

Tato diplomová práce se zabývá možnostmi pokročilého programování tohoto robotu v součinnosti se systémem rozpoznávání obrazu integrovaném v softwaru PickMaster 3. Tento software je určen pro rychlou konfiguraci linky i obsluhovaného sortimentu bez hlubších znalostí RAPID kódu. Ve své zamýšlené jednoduchosti však naráží na řadu mantinelů, které je nutno řešit přímo v RAPID kódu mimo tento program, na což není primárně uzpůsoben (zejména se jedná o editaci kódu v textovém editoru).

Hlavním cílem diplomové práce je příprava a realizace výukových úloh. Vedlejším cílem je vytvoření efektní interaktivní úlohy, která bude používána pro prezentace jak tohoto pracoviště, tak Katedry robototechniky a VŠB-TU Ostrava.

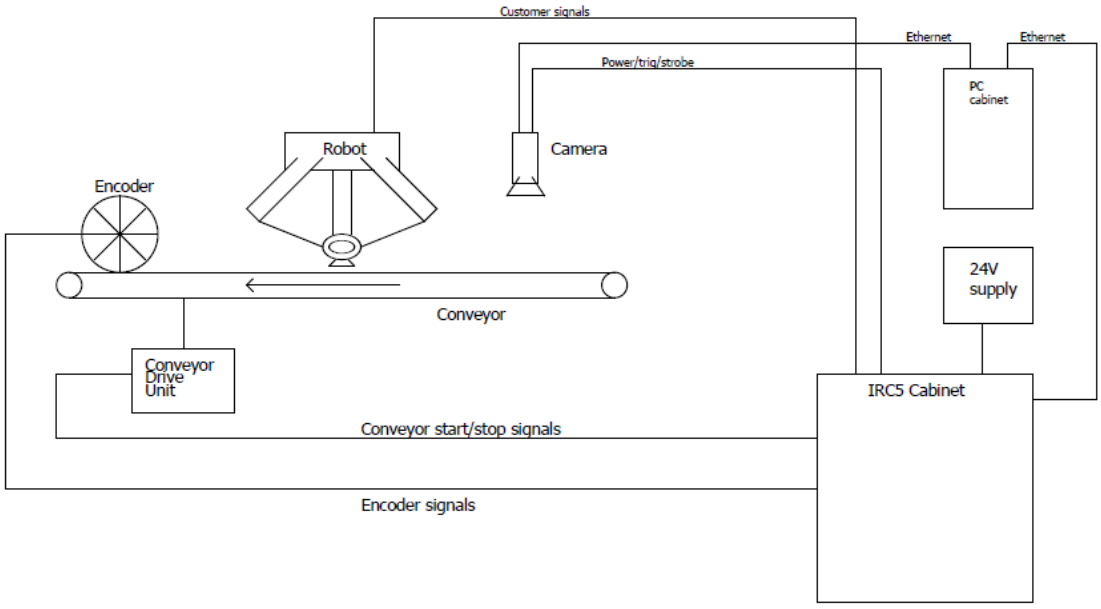
2 Cíle diplomové práce

1) Oživení pracoviště s robotem ABB IRB 360 na Centru robotiky
2) Instalace periférií a ovládacích prvků
3) Vytvoření stabilních světelných podmínek pro kameru (světelný box)
4) Analýza standardního programu aplikace PickMaster 3 a vstupu do něj za účelem uživatelských modifikací
5) Návrh demonstračních úloh, popis jejich programu a realizace za účelem budoucí výuky na pracovišti

3 Pracoviště s ABB IRB 360 FlexPicker

3.1 Pracoviště na centru robotiky

Schéma standardního pracoviště s jedním robotem, jednou kamerou a jedním dopravníkem, tedy shodného s pracovištěm nainstalovaným v laboratoři na centru robotiky (dále LCR), je zobrazeno na Obr. 1.



Obr. 1 Schéma pracoviště [3]

Jedním z hlavních prvků robotizovaného pracoviště je dopravník s šířkou pásu 450 mm a délkou 3030 mm, dodaný firmou MSV SYSTEMS CZ s.r.o..

Součástí konstrukce dopravníku je enkodér SICK DFS60B-S4PA10000 (viz Tab. 1) umístěný na konci dopravníku spolu se sestavou elektrického pohonu od firmy SEW-EURODRIVE CZ s.r.o. (viz Tab. 2).

Enkodér **SICK DFS60B-S4PA10000** [4]		
	Typ čidla	Inkrementální enkodér
	Vstupní napětí	5...32V
	Počet impulsů na otáčku	10000
	Stupeň krytí	IP65
	Připojení	Zásuvka M23, 12 pinů radiální

Tab. 1 Enkodér SICK DFS60B-S4PA10000

Spiroidní převodový motor **SEW EURODRIVE WA20/T DR63M4/TH**		
	Výkon	0,18 kW
	Krouticí moment	19 Nm
	Převodový poměr	i = 19,5
	Napájení	220-240 Δ / 380-415 Y
	Jmenovitý proud	0,87 A

Tab. 2 Spiroidní převodový motor SEW EURODRIVE WA20/T DR63M4/TH

Dopravník je spolu s robotem a kamerou integrován do rámu z hliníkových stavebnicových profilů (Obr. 2).

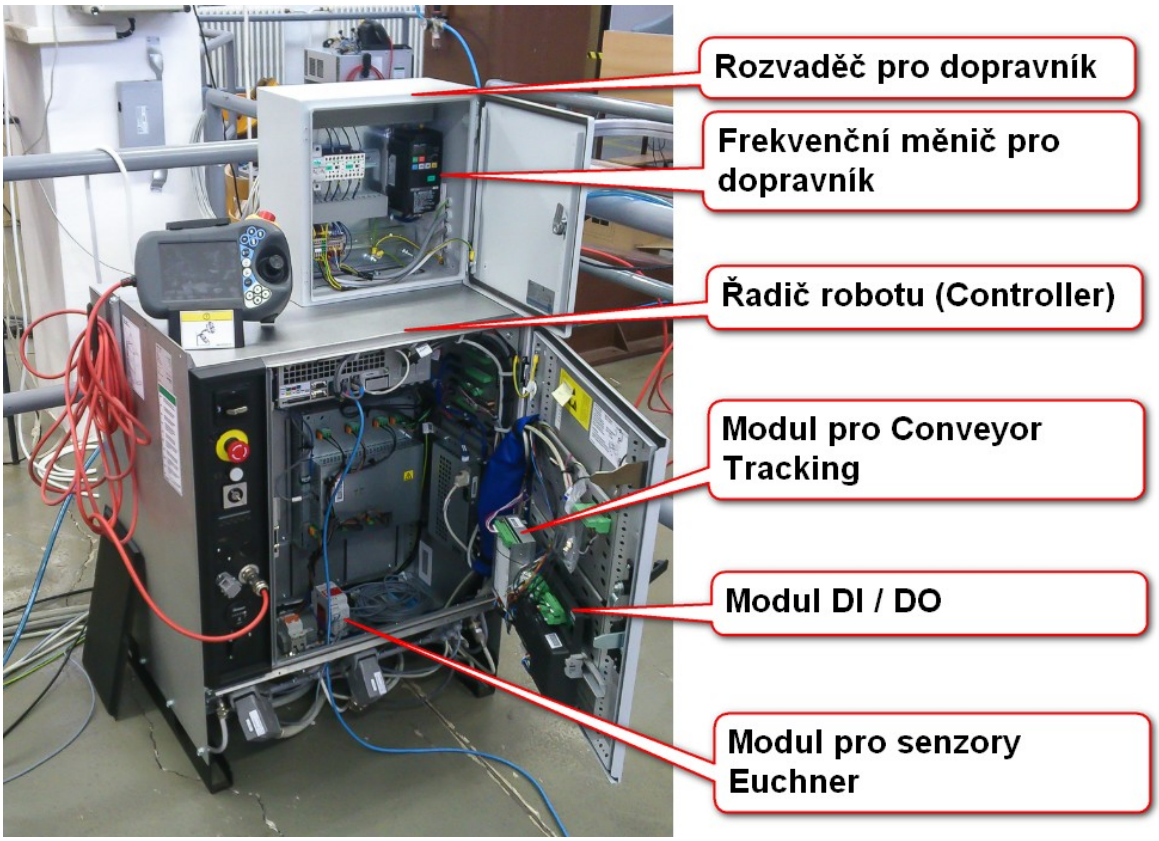
Obr. 2 Rám robotu a světelného boxu

Nad rámec základního osazení řadiče robotu (*Controlleru*) jsou na pracovišti na LCR umístěny následující komponenty (Obr. 3) :

- Rozvaděč a frekvenční měnič pro dopravník (Tab. 3)
- Modul pro sledování dopravníku (*Conveyor Tracking*)
- Modul vyhodnocovací jednotky pro senzory EUCHNER (viz níže)

Měnič **OMRON JXAB002** [5]		
	Napájení	1 x 230 VAC
	Připojitelný výkon motoru	0,2 kW
	Jmenovitý výstupní proud	1,4 A
	Maximální frekvence	400 Hz
	Metoda řízení	Fázová sinusová modulace PWM (V/f)

Tab. 3 Měnič OMRON JXAB002



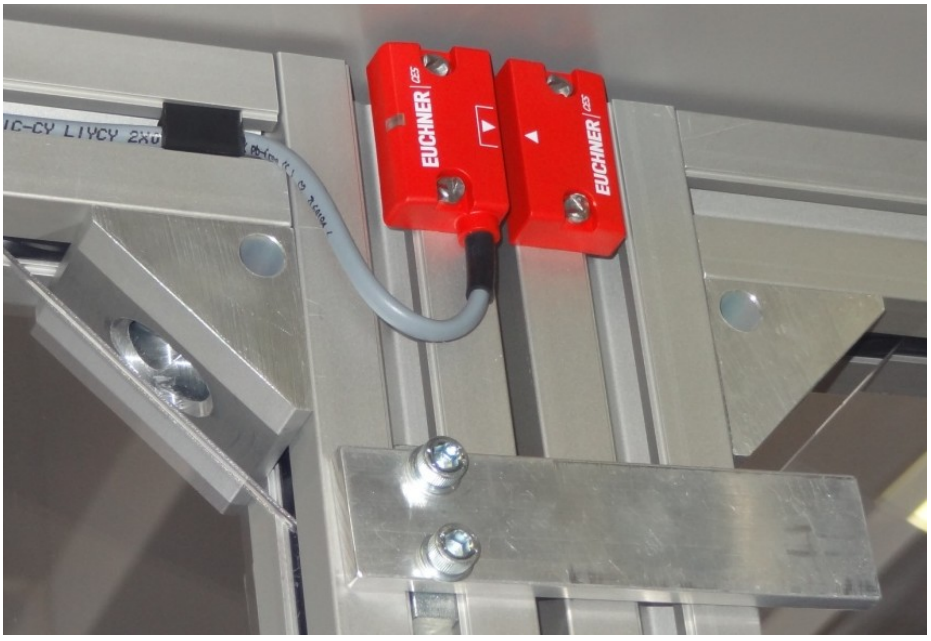
Obr. 3 Osazení řadiče robotu

Pro zajištění bezpečnosti na pracovišti jsou na rámech dveří instalovány bezkontaktní senzory Euchner CES-A-LLN (Obr. 4). Ty jsou připojeny do vyhodnocovací jednotky CES-AZ umístěné ve skříni řadiče robotu (Obr. 3).

Systémy CES se skládají z vyhodnocovací jednotky, čtecí hlavy a aktuátoru. Vyhodnocovací jednotka má analogové obvody schopné několikrát za sekundu číst přes anténu (čtecí hlavu) unikátní kód uložený v RFID transpondéru, který je zapouzdřen uvnitř aktuátoru. Pokud dojde ke správnému přečtení kódu, znamená to, že jsou dveře/kryty zavřeny a jednotka sepne svoje bezpečnostní výstupy.[6]

Další vlastnosti bezkontaktního bezpečnostního systému řady CES ([6], [7]):

- Plně kódovaný systém- každý aktuátor (transpondér) je originál, poskytuje tak nejvyšší možnou úroveň ochrany proti neoprávněné manipulaci.
- Velký čtecí dosah, není nutné přesně seřizovat dveře. I nepřesné seřízení dveří nebo dokonce seřízení přesně na hranici zajištěného čtecího dosahu nezpůsobí nechtěné vypnutí.



Obr. 4 Čtecí hlava a aktuátor Euchner na rámu dveří

Další informace o pracovišti a instalovaných periferiích v kapitole 4.

3.2 ABB IRB 360 FlexPicker

Je již druhou generací robotu typu delta, vyráběného švédskou firmou ABB. Tento nový robot navazuje na široce využívaného předchůdce ABB FlexPicker IRB 340. Je výsledkem desetiletého vývoje a testování s využitím praktických zkušeností v oboru balicí techniky. Firma ABB má v různých zemích světa více než 1800 instalací delta robotů a je špičkou v oboru manipulační a balicí techniky [8].

Na rozdíl od univerzálních angulárních robotů jsou roboty typu delta specializované pro aplikace odebírání a přemisťování předmětů vysokou rychlostí. Kinematika těchto robotů umožňuje dosáhnout velkých změn polohy koncového bodu (TCP) při minimálních změnách polohy poháněných kloubů. Vysoká rychlost je dále možná díky lehké pohybující se části skeletu. Maximální rychlost TCP je 10000 mm/s. Této rychlosti však většinou nedosahuje z důvodu převážně kratších pohybů TCP a omezeného zrychlení (rampy).

Obr. 5 IRB 360 FlexPicker [8]

Tyto roboty nalézají uplatnění nejčastěji v potravinářství, proto musí splňovat nejpřísnější hygienické podmínky. Lze objednat v nerezové verzi s krytím IP69K, kterou je možno omývat horkou tlakovou vodou a průmyslovými detergenty (Obr. 6). Tato verze je nutností pro závody zpracující maso a mlékárenské výrobky [9].

ABB nabízí čtyři varianty [8]:

- Kompaktní verze: IRB 360-1/800 – dosah 800mm, nosnost 1kg
- Standardní verze: IRB 360-1/1130 – dosah 1130mm, nosnost 1kg
- Verze s vysokou nosností: IRB 360-3/1130 – dosah 1130mm, nosnost 3kg
- Verze „dlouhé rameno“: IRB 360-1/1600 – dosah 1600mm, nosnost 1kg

Obr. 6 Omývání robotu tlakovou vodou [10]

Vybrané technické parametry jsou uvedeny v Tab. 4.

ABB IRB 360-3/1130 [9]	
Nosnost	3 kg
Průměr prac. prostoru	1130 mm
Počet os	3/4 (dodaný na LCR má 4 osy)
Opakovatelná přesnost[1]	0,1 mm
Integrovaný zdroj signálů	12 signálů, max 50V, 250 mA
Integrovaný zdroj vakua	Přetlak max 7 bar / podtlak max 0,75 bar
Provozní teplota okolí	0°C až 45°C
Hmotnost	120-145 kg
Napájení	200-600 V
Jmenovitý výkon transformátoru	7,2 kVA
Spotřeba energie při typickém cyklu	0,477 kW při zátěži 1 kg

Tab. 4 Technické parametry IRB 360-1/1130

[1] Absolutní přesnost závisí na kvalitě kalibrace soustavy kamera- dopravník- robot

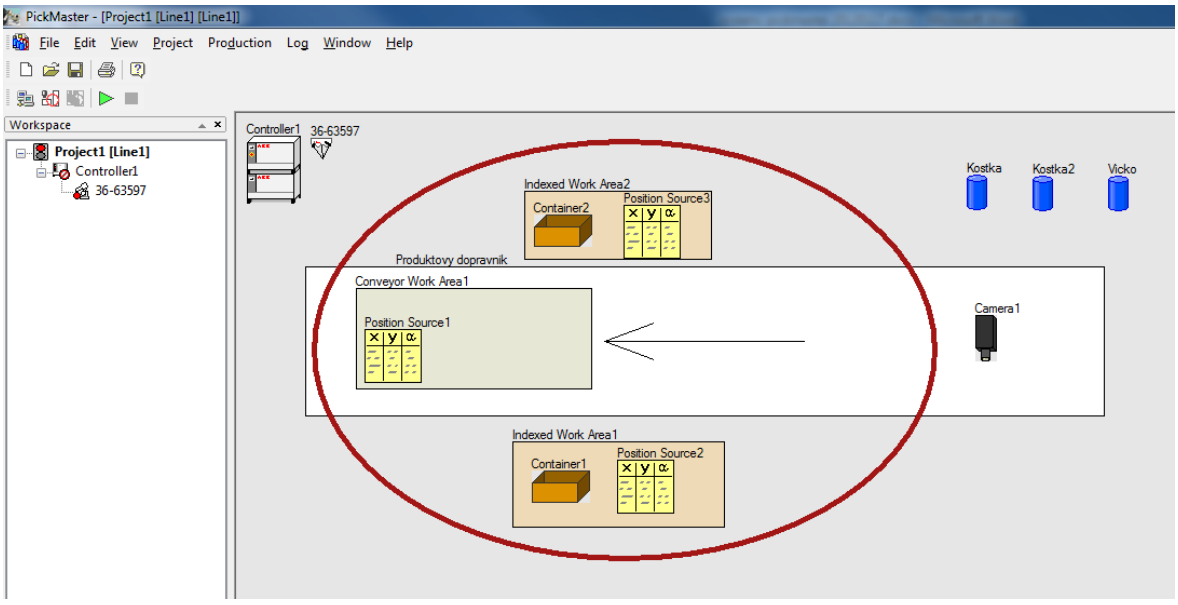
3.3 PickMaster 3

Systém FlexPicker se skládá z robotu ABB IRB 360, jeho řídicího systému (řadiče robotu - *Controlleru*), systému sledování dopravníku (*Conveyor Tracking*) a specializovaného softwaru PickMaster 3.

Tento softwarový produkt používá jednoduché grafické rozhraní pro rychlou konfiguraci balicích aplikací, ve kterých může podél dopravních pásů pracovat současně až osm robotů. Obsahuje výkonný systém strojového vidění a schopnost sledování pásu. Zároveň je však otevřený vůči komunikaci s dalšími externími senzory [11].

Idea je taková, že i málo zkušený uživatel je schopen nakonfigurovat jednoduchou linku i nový sortiment v řádu několika hodin. Při tom nemusí umět programovat, či znát RAPID kód. Konfigurace je prováděna ve schematickém grafickém prostředí (Obr. 7).

S touto jednoduchostí však jde ruku v ruce fakt, že toto řešení nelze nazvat silným. Pokud chce uživatel provést pokročilé operace, kde pokročilou operací je myšleno už třídění do dvou různých zásobníků, je nutno editovat samotný RAPID kód. To je pro uživatele, respektive nyní už programátora, zřejmě největší bolestí PickMasteru. Více o možnostech editace v kapitole 5.4.



Obr. 7 Příklad vzhledu linky v prostředí PickMaster 3

4 Instalované periferie

Pro realizaci všech zamýšlených úloh a budoucí výuku na pracovišti byly navrženy dodatečné periferie. Následuje jejich stručný výčet s odkazy na příslušné podkapitoly:

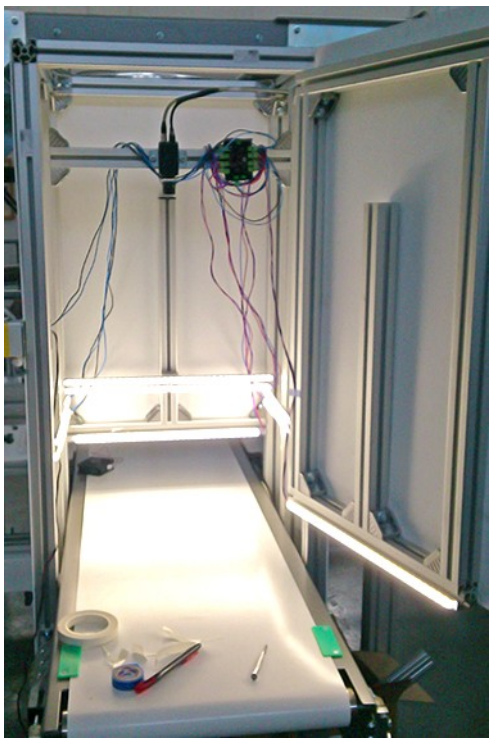
- Světelný box (*Softbox*) - kapitola 4.1
- Odkládací stoly - kapitola 4.1
- Panel ovládání pracovních prostorů - kapitola 4.2
- Bluetooth modul - kapitola 4.3
- Matrice pro úlohu skládání znaků - kapitola 6.3
- Vakuová přísavka

Poté byly jednotlivé komponenty poptány, nakoupeny a sestaveny ve spolupráci s řemeslnou dílnou na Centru robotiky. Základními prvky jsou hliníkové stavebnicové profily Bosch (kompatibilní s konstrukcí rámu) a desky z pěněného PVC, které byly řezány vodním paprskem ve spolupráci s Hornicko-geologickou fakultou VŠB-TU Ostrava. Dále byla pro účely uchopování malých předmětů zakoupena hluboká vakuová přísavka průměru 10 mm a redukce na stávající šroubení.

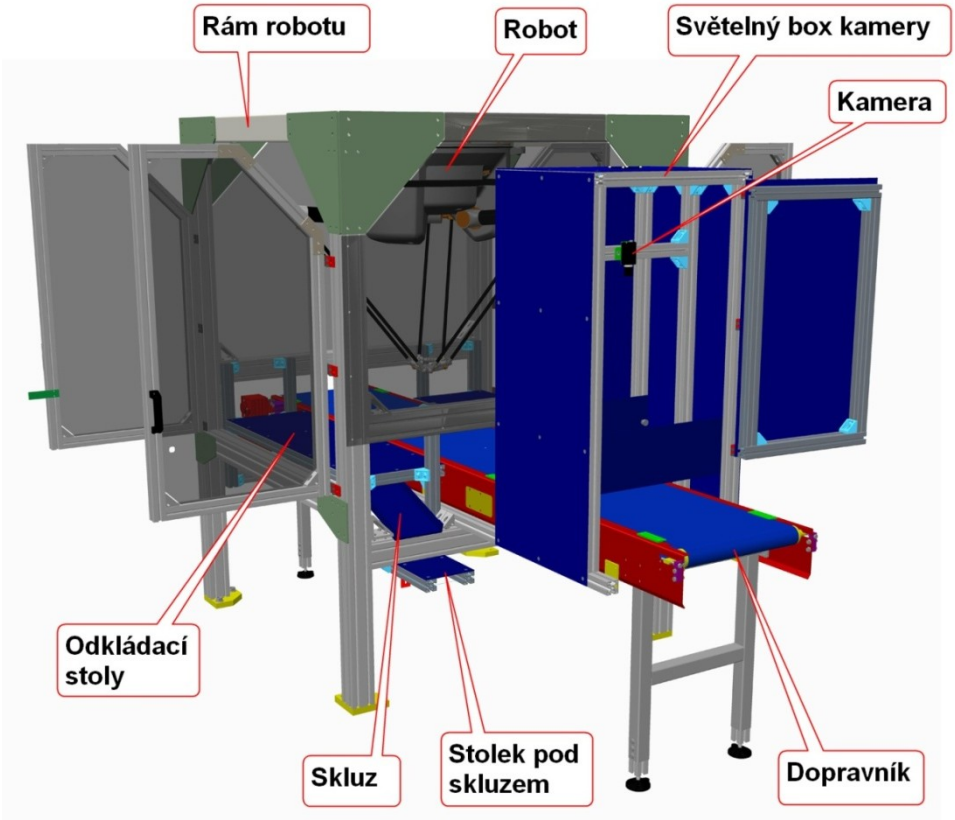
4.1 Světelný box a odkládací stoly

Při rozpoznávání obrazu jsou vždy velmi důležité světelné podmínky. Jejich stabilizace se ukázala jako nezbytná, zejména jsou-li rozpoznány předměty podle barvy, například žetony v úloze v kapitole 6.2. Proto byl navržen a sestaven světelný box (*softbox*). Uvnitř je instalováno osm osvětlovacích segmentů - LED pásek v hliníkovém rámečku s difuzorem (Obr. 8). Díky tomu bylo docíleno stabilních světelných podmínek, rovnoměrného nasvícení rozpoznávaných objektů manipulace a konstantního vyvážení bílé.

Dále byly navrženy a namontovány postranní odkládací plochy - výškově stavitelné stolky a skluz pro dopravu předmětu ven z pracovního prostoru robotu (Obr. 9).



Obr. 8 Světelný box

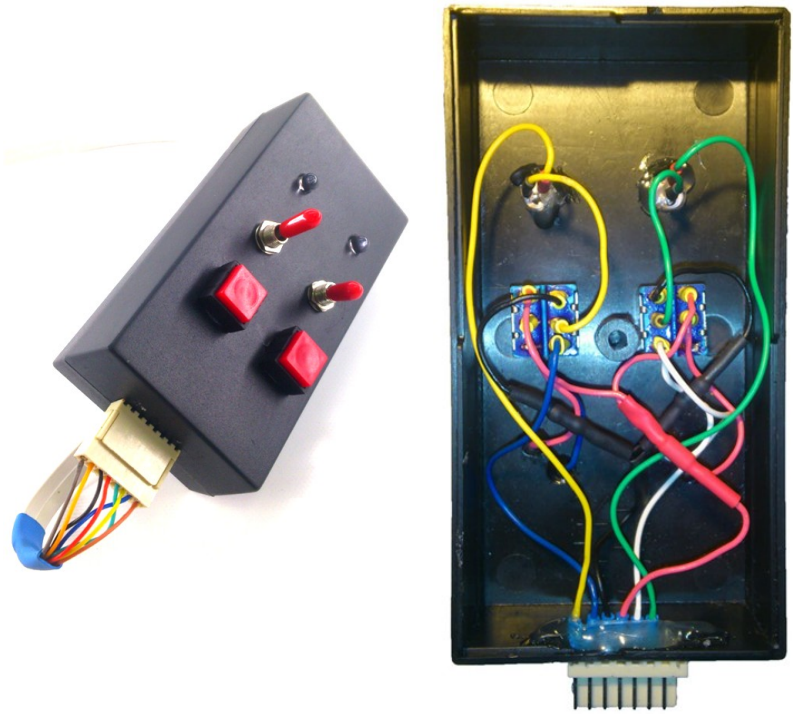


Obr. 9 CAD 3D návrh konstrukce periferií

4.2 Panel ovládání pracovních prostorů

Při reálné aplikaci robot většinou odkládá objekty manipulace (OM) do připraveného zásobníku, který má konečnou kapacitu, nebo na další dopravník, který má konečnou rychlost. Pokud se zásobník zaplní, nebo dopravník není schopen odsouvat OM dostatečně rychle, robot pozastaví činnost. Po výměně, zásobníku nebo uvolnění dopravníku robot pokračuje. Tyto stavy jsou sledovány pomocí signálů, například z čidla přítomnosti zásobníku.

Protože pracoviště na LCR systém sledování zásobníků neobsahuje, je tato funkce nahrazena panelem ovládání pracovních prostorů (Obr. 10).



Obr. 10 Panel ovládání pracovních prostorů a jeho vnitřní zapojení

4.2.1 Funkce a vnitřní elektrické zapojení

Zaplní-li se pracovní prostor 1 (*"Work Area 1"*, dále *"WA 1"*), řadič robotu vyšle signál, který je přiveden na vstup panelu *"Signál plného WA 1"* a čeká na *"Signál* potvrzení *WA 1"*, který je přiveden zpět do řadiče. Ovladač pak může fungovat ve dvou režimech, které se přepínají páčkovým přepínačem *"Přepínač cross connection WA 1"* [2]. Obdobně pro *"WA 2"*.

1. Režim manuálního ovládání pracovních prostorů (přepínač v poloze zobrazené na schématu zapojení):

Příchozí signál *"Signál plného WA 1"* rozsvítí *"LED plného WA 1"*, protože je obvod uzavřen do *"GND"*. Napájení 24V je *"Přepínačem cross connection WA 1"* připojeno na *"Tlačítko potvrzení WA 1"*. Po stisknutí tohoto tlačítka je tak vyslán *"Signál potvrzení WA 1"*. Robot pak pokračuje v odkládání do daného pracovního prostoru a LED zhasne. Obdobně pro WA 2.

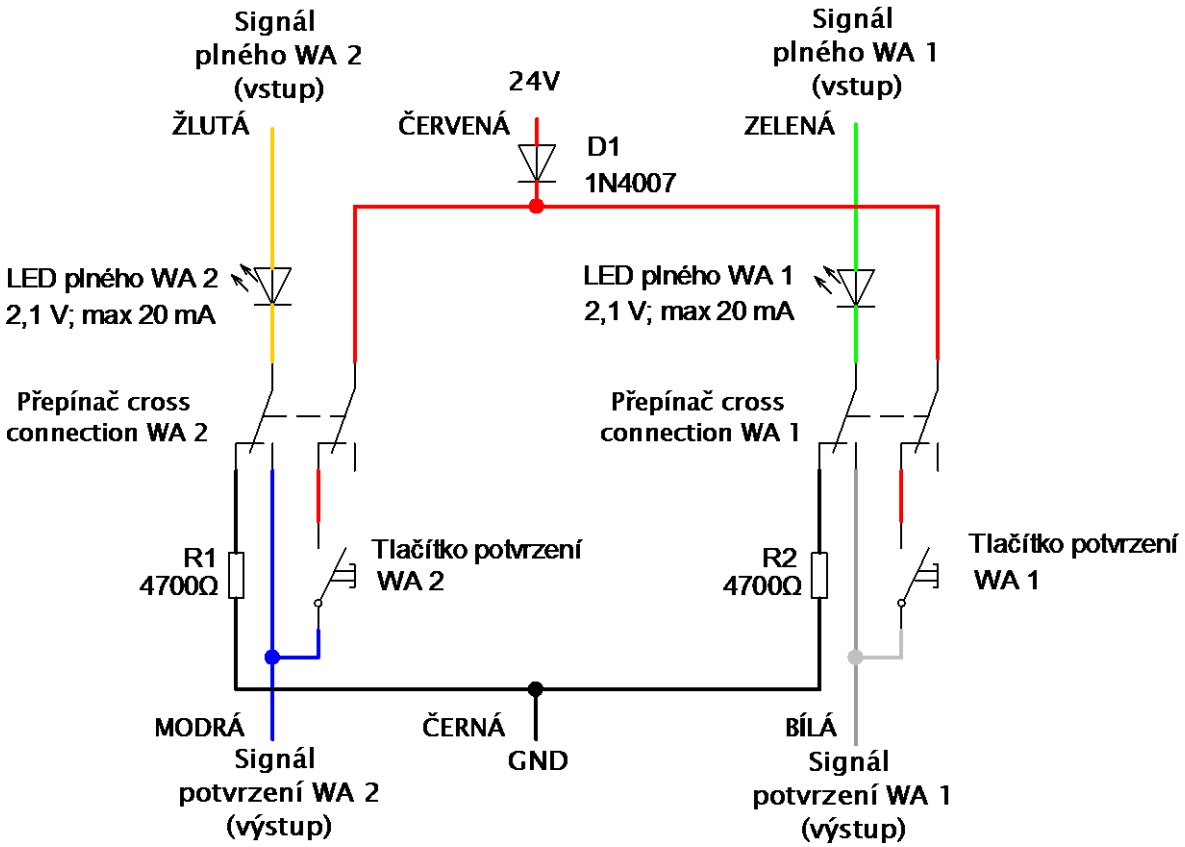
2. Režim automatického potvrzení - nekonečný zásobník (přepínač spojuje pravé větve kontaktů):

Příchozí signál *"Signál plného WA 1"* je ihned křížově spojen (anglicky cross connected) na *"Signál potvrzení WA 1"*. Robot tak pokračuje v odkládání do daného pracovního prostoru bez čekání. V tomto režimu je zároveň na přepínači odpojeno napájení tlačítka.

Obvod je navržen pro proud 5mA. Průchod proudu ze vstupního signálu přímo do *"GND"* je omezen rezistory R1, resp. R2 (4700Ω) a LED (105Ω), celkem tedy 4805Ω. Průchod proudu ze zdroje 24V do výstupních signálů je v ovládacím panelu omezen pouze LED (105Ω). Je však dále omezen interním rezistorem na vstupu DI desky řadiče navrženým právě pro proud 5mA při vstupním napětí 24V (tedy s hodnotou odporu 4800Ω) [12]. Svítivost LED při proudu 5mA (maximum pro použitý typ je 20mA) byla předem úspěšně otestována v nepájivém poli.

Výpočty byly provedeny dle Ohmova zákona $U = R * I$.

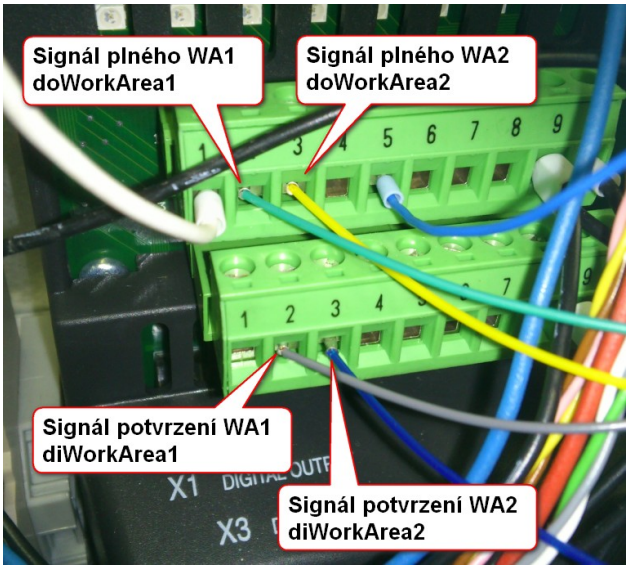
[2] Anglický výraz *Cross connection* byl zvolen proto, že přepínač mechanicky supluje stejnojmennou funkci softwarového křížového spojení signálů v řadiči.



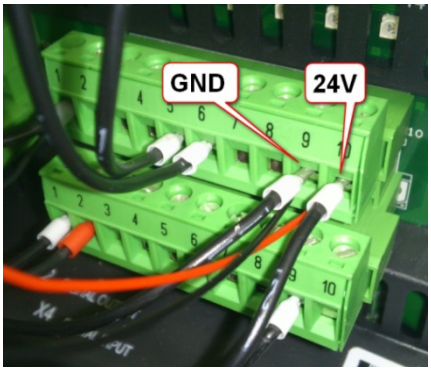
Obr. 11 Schéma elektrického zapojení panelu ovládání pracovních prostorů

4.2.2 Připojení k řadiči a konfigurace

Do řadiče robotu byl přiveden kabel ovládacího panelu a jeho jednotlivé vodiče zapojeny do I/O desky dle Obr. 12 a Obr. 13. Poté byly v konfiguraci signálů řadiče vytvořeny příslušné signály, viz Obr. 14.



Obr. 12 Zapojení signálů ovládacího panelu



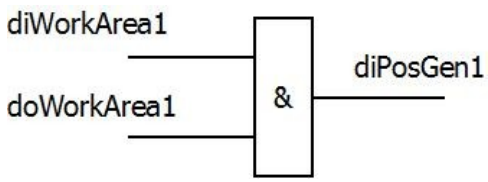
Obr. 13 Zapojení napájení ovládacího panelu

di_Vacuum1	Digital Input	BOARD11		8
diPosGen1	Digital Input	PPASIM		1
diPosGen2	Digital Input	PPASIM		2
diStrobeSignal	Digital Input	PPASIM		6
diWorkArea1	Digital Input	BOARD11		1
diWorkArea2	Digital Input	BOARD11		2
doBlow1	Digital Output	BOARD11		13
doManSync1	Digital Output	BOARD11		8
doManSync2	Digital Output	BOARD11		9
doManSync3	Digital Output	BOARD11		10
doManSync4	Digital Output	BOARD11		11
doResetEstop	Digital Output	PPASIM		0
doStartCnv1	Digital Output	BOARD11		0
doTrigSignal	Digital Output	PPASIM		6
doTrigVis1	Digital Output	BOARD11		4
doTrigVis2	Digital Output	BOARD11		5
doTrigVis3	Digital Output	BOARD11		6
doTrigVis4	Digital Output	BOARD11		7
doVacuum1	Digital Output	BOARD11		12
doWorkArea1	Digital Output	BOARD11		1
doWorkArea2	Digital Output	BOARD11		2
DRV1BRAKE	Digital Output	DRV_1	Brake-release coil	2

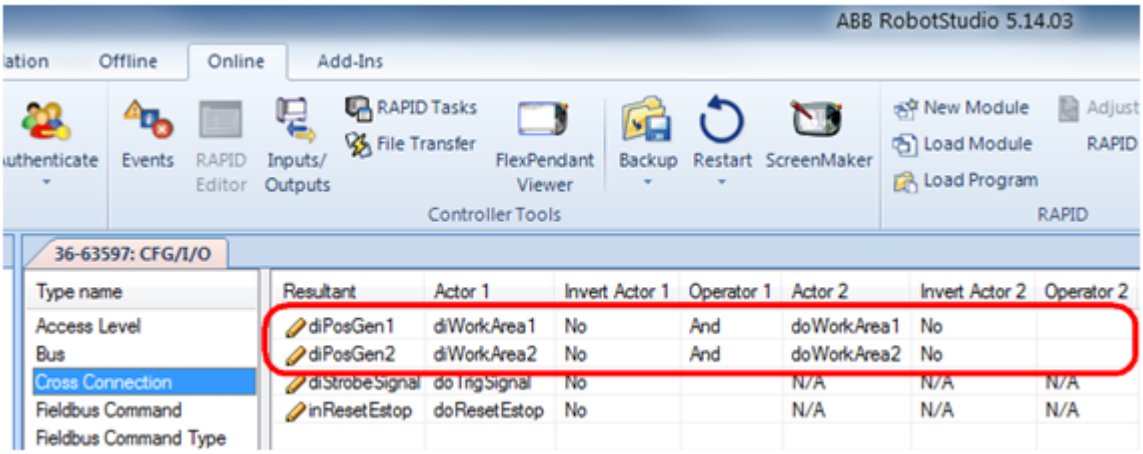
Obr. 14 Vytvořené signály pro ovládací panel

Kromě fyzických signálů definovaných přímo na I/O desce byly vytvořeny navíc dva simulované signály *diPosGen1* a *diPosGen2*. Vyvstal totiž problém, že robot si signály potvrzení nových zásobníků hromadil (střádal do zásoby) a to i tehdy, když nový zásobník nepotřeboval. Pokud se během naplňování jednoho zásobníku stisklo tlačítko potvrzení WA, robot pokračoval v plnění daného zásobníku znovu okamžitě po jeho naplnění, tedy bez dalšího dotázání. Tento problém byl alespoň částečně odstraněn vytvořením vnitřní logiky tak, aby robot přijímal signál potvrzení nového WA jen když jej potřebuje.

Logický součet dle Obr. 15 byl vytvořen pomocí funkce *Cross Connection,* viz Obr. 16.

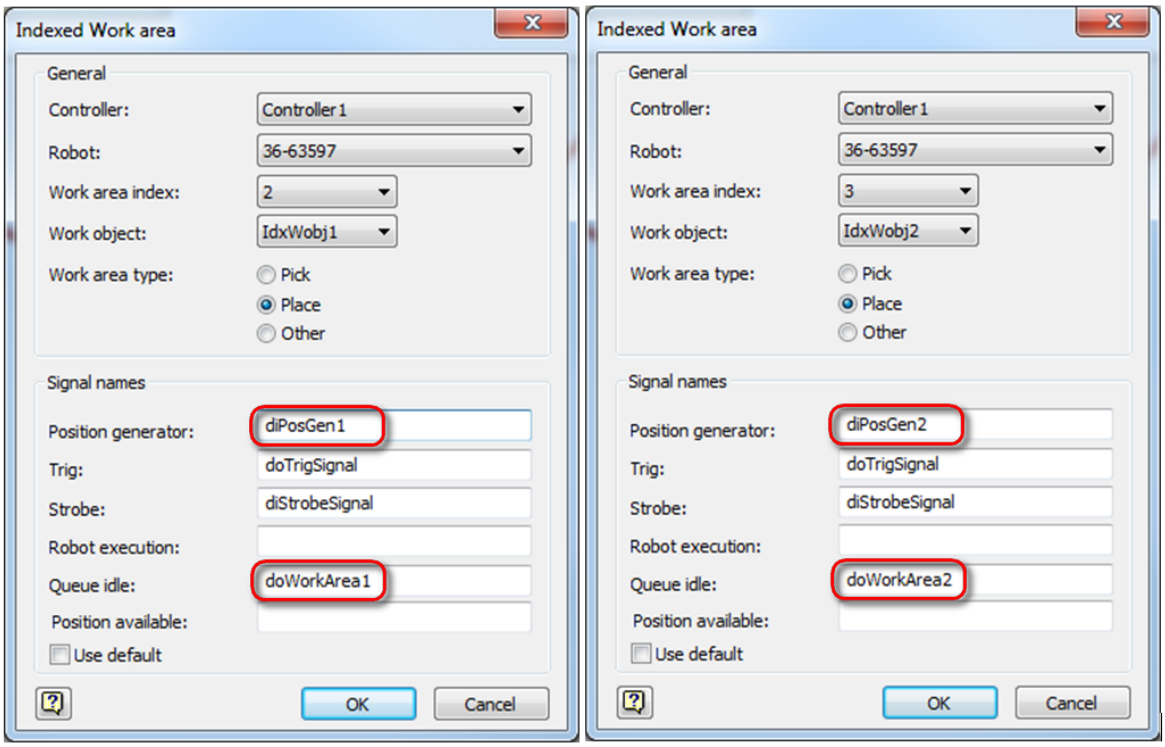


Obr. 15 Logický součet signálů pracovních prostorů



Obr. 16 Nastavení Cross Connection signálů pracovních prostorů

V robotu je tedy již definováno čtení i posílání signálů. Dále byly nastaveny signály v PickMasteru. Do nastavení signálu se lze dostat po dvojkliku na příslušný pracovní prostor (*Work Area*) v prostředí konfigurace linky (*Line*). Signály by měly být vyplněny dle Obr. 17.



Obr. 17 Nastavení signálů pracovních prostorů v prostředí PickMaster

4.3 Bluetooth modul

Pro vstup do řadiče robotu a probíhajícího programu, zejména v úloze skládání znaků do matrice (kap. 6.3) byl vybrán, poptán a zakoupen Bluetooth modul connectBlue OBS433i. Jeho vybrané parametry jsou uvedeny v Tab. 5.

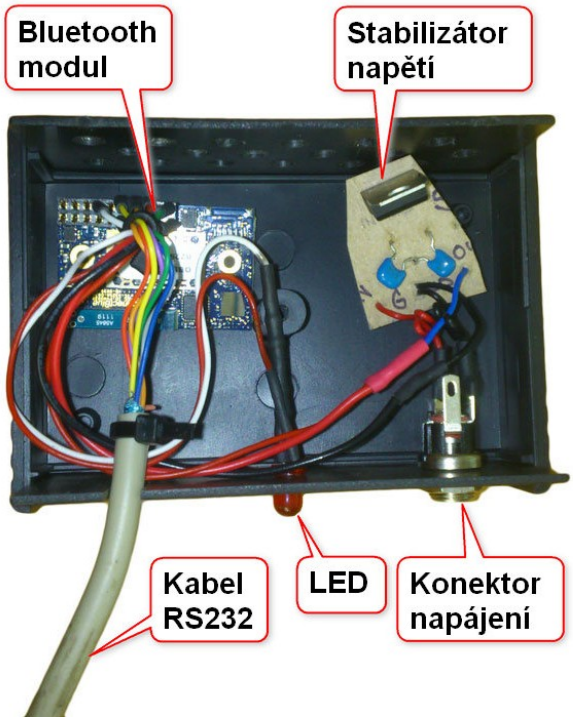
connectBlue OBS433i Bluetooth modul s interní anténou a 20-ti pinovým konektorem		
	Rozměry	23 x 36 x 4 mm
	Teplotní rozsah	-40°C až +85°C
	Výstupní výkon	1,9 dBi (interní anténa)
	Dosah	1000m
	Baudrate	1200 bit/s – 1,8 Mbit/s
	Napájecí napětí	3,3 – 6,0 V
	Spotřeba	50 mA

Tab. 5 Bluetooth modul OBS433i [13]

4.3.1 Elektrické zapojení

Byl zakoupen a upraven plastový box, do kterého je umístěn samotný Bluetooth modul, stabilizátor napětí (viz níže), konektor napájení a LED signalizující zapnutí modulu (tlumený svit) a připojení modulu ke vzdálenému zařízení (plný svit).

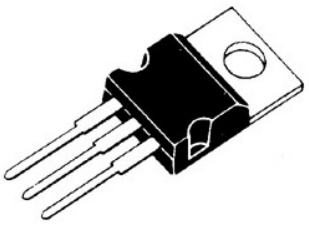
Pohled do osazeného boxu je na Obr. 18.



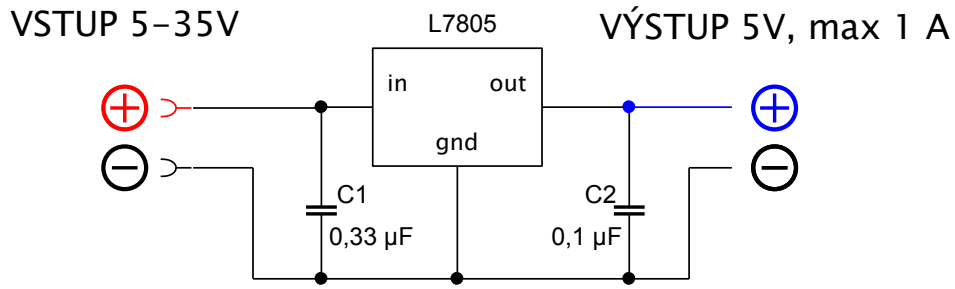
Obr. 18 Sestava Bluetooth modulu s napájením a LED

Stabilizátor napětí

Vzhledem k momentální nedostupnosti zdroje napětí 3,3 – 6V a nízkým nárokům modulu na napájecí proud byl vyroben jednoduchý stabilizátor napětí 5V. Skládá se ze stabilizátoru 7805 (vybrané parametry viz Tab. 6) a dvou kondenzátorů. Schéma zapojení je na Obr. 19. Vstupní vodiče jsou připojeny na konektor napájení, výstupní vodiče na patici konektoru Bluetooth modulu.

7805 TO220 Lineární regulátor napětí 5V/1A, pouzdro TO220		
	Výstupní napětí	5 V
	Výstupní proud	1 A
	Max vstupní napětí	35 V
	Úbytek napětí	2 V

Tab. 6 Lineární regulátor napětí 7805 [14]

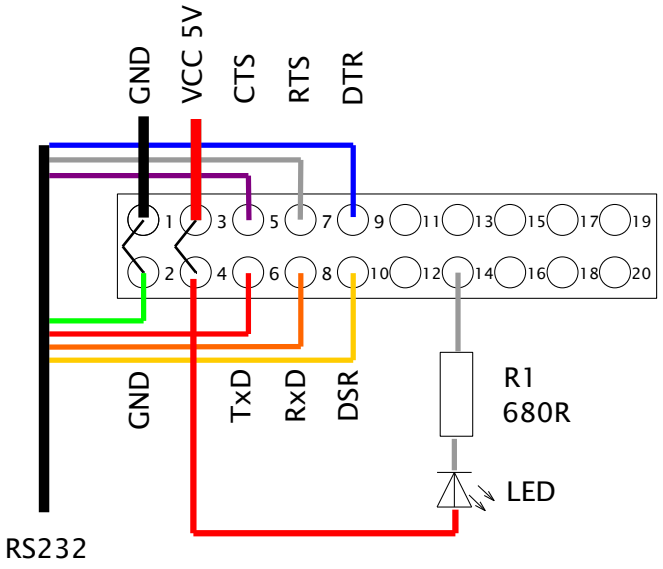


Obr. 19 Elektrické schéma stabilizátoru napětí

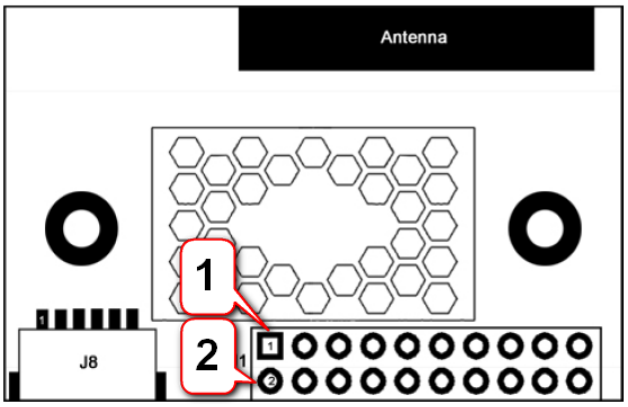
Konektor Bluetooth modulu

Modul podporuje připojení přímo na RS232 s úrovněmi 5V, proto může být kabel RS232 připojen přes patici přímo na modul. Schéma připojení podle barev žil je na Obr. 20. Do patice je připojena i LED signalizace připojení ke vzdálenému zařízení.

Pro připojení je použita dutinková lišta rovná, 2 řady, rozteč 2mm, 2x10 pinů (například BTK2X10G zde: http://www.gme.cz/dutinkove-listy/dutinkova-lista-ostatni-btk2x10g-p832-124/). Zapojení je provedeno dle [15].



Obr. 20 Schéma zapojení konektoru Bluetooth modulu



Obr. 21 Schéma pinů na Bluetooth modulu OBS433i [15]

4.3.2 Konfigurace modulu

Zdroje a manuály:

Bluetooth Serial Port Adapter Toolbox - Getting Started [16]
- popis konfigurace BT modulu přes sériový port a aplikace SPA Toolbox

Bluetooth Serial Port Adapter Security [17]
- popis možností nastavení zabezpečení

Postup konfigurace:

Připojení BT modulu přes sériový port k PC a jeho konfigurace pro spárování s mobilním telefonem je popsáno v příloze [Příloha A].

4.3.3 Sériová komunikace s řadičem robotu

Po připojení Bluetooth modulu k řadiči robotu je nutno napsat obslužné rutiny. Zde nastaly a následně byly vyřešeny tři problémy:

a) Ošetření nechtěně odeslaných zpráv z mobilního telefonu
b) Neočekávaný počet znaků v příchozí zprávě a chyba v odesílání zprávy
c) Čtení příchozích zpráv proměnné délky

ad a) **Ošetření nechtěně odeslaných zpráv z mobilu**

Při psaní zprávy na mobilním telefonu se může lehce stát, že zprávu uživatel pošle ještě před jejím dokončením, například přehmátnutím. Zde je použita metoda, kdy musí být odesílaná zprávy z obou stran ohraničena znakem tečky. Tím dá uživatel při psaní zprávy jasně najevo, že zprávu ukončil. Odesílaná zpráva tak může vypadat například takto:

.ZPRAVAKODESLANI.

Alternativně lze i na mobilním telefonu realizovat potvrzovací dialog, například "Opravdu chcete napsat...?" s následným požadavkem na potvrzení znakem "A" (ano) nebo "N" (ne). Tato funkce zde však není vytvořena.

ad b) **Neočekávaný počet znaků v příchozí zprávě a chyba v odesílání zprávy**

Při ladění rutiny čtení se ukázalo, že délka příchozí zprávy zjištěná funkcí *StrLen* neodpovídá očekávané délce a je vždy o 2 znaky delší. Například, pokud byla z mobilu odeslána zpráva ".A." (tedy o délce 3 znaků), v robotu se zpráva zobrazila správně (tedy opět 3 znaky), ale délka zprávy zjištěná funkcí *StrLen* byla 5 znaků.

Při odesílání zprávy z mobilu se totiž na konec zprávy přidává netisknutelný znak *Carriage Return* (CR), který posouvá kurzor na začátek řádku. ASCII kód tohoto znaku je 0x0D (hexadecimálně) [18]. Systém robotu tento netisknutelný znak čte jako dva znaky, proto je délka zprávy o tyto 2 znaky delší, i když se při vypsání příchozí zprávy (například na FlexPendant) zobrazí zpráva bez těchto dvou znaků. S tímto je nutno počítat při dalším zpracování zprávy.

Fakt, že u příchozí zprávy je na konci netisknutelný znak *CR* byl tedy odhalen, dále však bylo nutno zjistit, jak jej z robotu odeslat. Aplikace v mobilu totiž tento znak taktéž potřebuje, aby "věděla", že příchozí zpráva je kompletní a má ji zobrazit. Jelikož je CR netisknutelný, není jej tak možné ani jednoduše napsat.

RAPID kód je však i na tuto eventualitu připraven. Hexadecimální zápis ASCII znaku lze jednoduše napsat za obrácené lomítko. Zápis *CR* je tedy *"\0D"*. Syntaxe při odesílání pak vypadá takto:

```
WriteStrBin channel, "zprava" + "\0D";
```

ad c) **Čtení příchozích zpráv proměnné délky**

Obecně se lze ve většině programovacích jazyků (RAPID, Wiring, C#, ...) setkat s problémem při čtení zpráv proměnné délky, například ze sériové linky. Syntaxe příkazu čtení totiž požaduje zadání délky zprávy, kterou má přečíst. V RAPIDu se jedná o příkaz:

```
ReadStrBin(channel,delkaZpravy \Time:=maxCas);
```

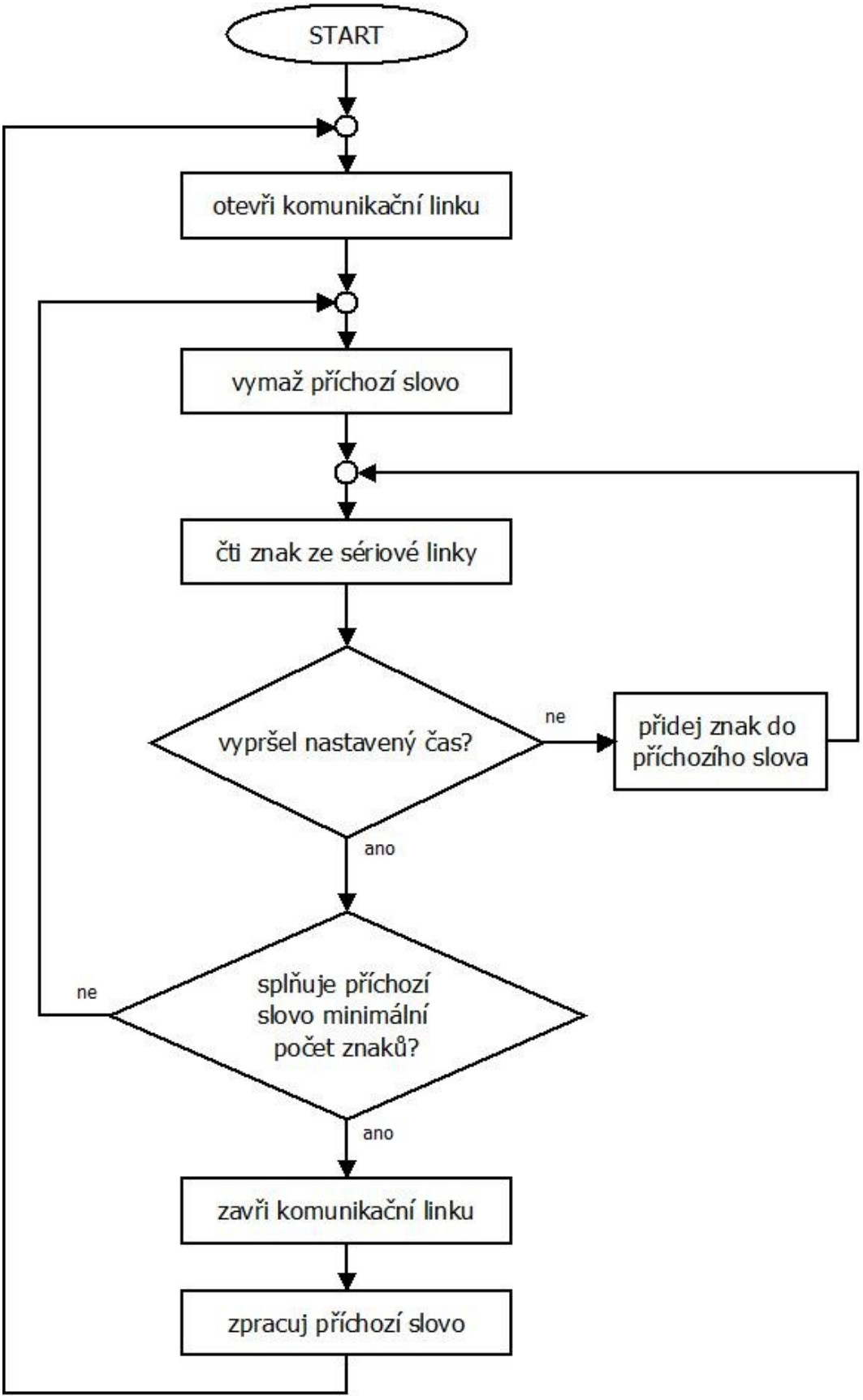
Pokud existuje možnost upravit syntaxi odchozí zprávy ze zařízení, se kterým komunikujeme (například z Arduina), lze odchozí zprávu ještě před odesláním upravit tak, aby měla vždy konstantní délku. Například, je-li odesílána jako text (*String*) s číselnou hodnotou v rozmezí 0 až 1023, tedy délky 1 až 4 znaků, je možno před odesláním k hodnotě přičíst 1000. Pak je odesílána (a přijímána) hodnota 1000 až 2023, tedy vždy 4 znaky. Při zpracování zprávy při příjmu se pak jednoduše 1000 opět odečte a získá se tak původní hodnota. Obdobně by se dalo postupovat i při posílání textu- znaky, které do počtu chybí by se před odesláním doplnily například mezerami.

Zde je však na straně odesílatele program Bluetooth Terminal (pro OS Android) a délku zprávy tak před odesláním automaticky upravit nelze. V tomto případě se nabízí dvě možnosti. Buďto uživatel odešle vždy zprávu s konstantním počtem znaků a nevyužité znaky doplní například mezerami, nebo využijeme funkcionality RAPID kódu popsané dále.

Příkaz *ReadStrBin* umožňuje nastavení maximálního času čtení (naslouchání sériové linky), po jehož vypršení zahlásí chybu (*ERROR*), která způsobí zastavení programu, není-li ošetřena. Výše v textu jde o parametr `\Time:=maxCas.`

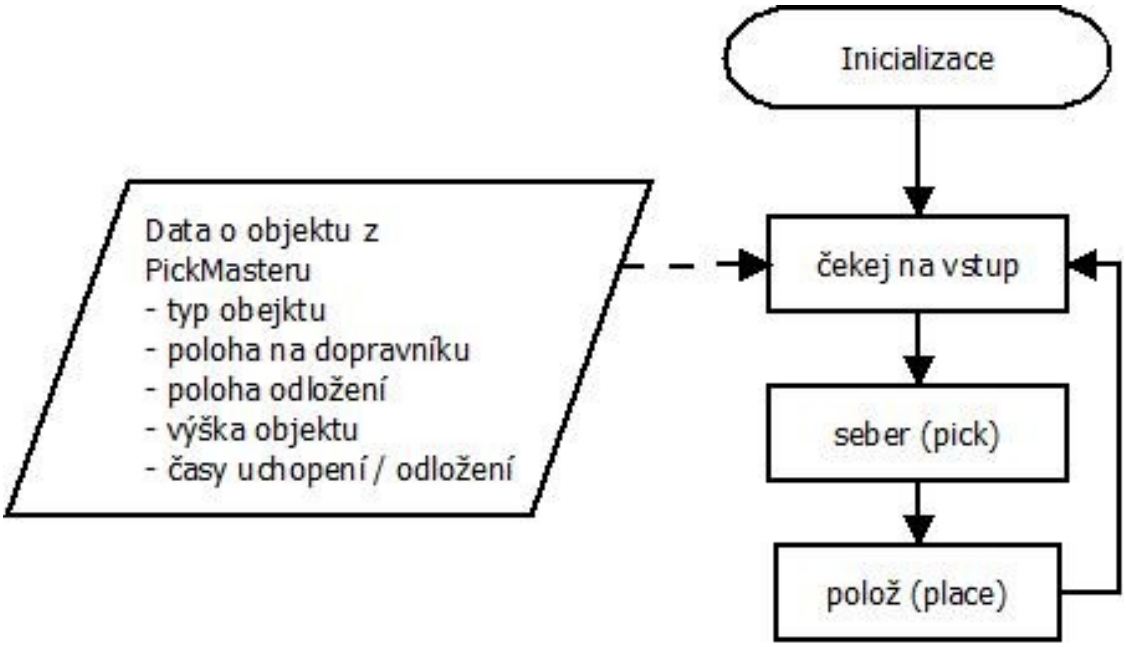
Algoritmus následujícího programu je založen na čtení jednotlivých znaků a faktu, že pokud po sériové lince přichází zpráva, její jednotlivé znaky (*serialCharacter*) přicházejí v minimálních časových intervalech za sebou. Pokud se tedy na lince nějaké znaky objeví, program je přidávák finální příchozí zprávě (*serialMessage*), dokud se znaky objevovat nepřestanou. Jakmile už po daný maximální časový interval *(*`\Time:=0.5`) nic nepřišlo, aktivuje se ošetření chyby "ERR_DEV_MAXTIME", které předává celou dosud přečtenou zprávu (*serialMessage*) k dalšímu zpracování rutině *serialRead*. Pokud za tento časový interval nepřišlo nic, obsahuje finální zpráva (*serialMessage*) prázdný textový řetězec (""), který nevyhoví kritériu minimální délky zprávy ve zpracující rutině (*serialRead*) a program se vrací zpět k čekání na příchozí komunikaci.

```
PROC serialRead ()
!Otevreni seriove komunikace
...
next1:
characterRead;
!Nasleduje kontrola syntaxe prichozi zpravy
...
!Pokud prichozi zprava NENI platna
GOTO next1;
!Pokud JE platna -> robot napise slovo
...
ENDPROC

PROC characterRead()
!nuluje pocitadlo znaku
characterCounter:=0;
!nuluje prichozi zpravu
serialMessage:="";

next2:
!nacita jeden prichozi znak
!pokud vyprsi timeout \Time, vyhazuje chybu ERR_DEV_MAXTIME
!chyba (ERROR) se obsluhuje na konci rutiny
serialCharacter:=ReadStrBin(channel,1 \Time:=0.5);
WaitTime 0.1;
!prida prichozi znak k prichozi zprave
serialMessage:=serialMessage+serialCharacter;
!inkrementuje pocitadlo znaku
Incr characterCounter;
!navrat zpet ke cteni dalsiho znaku
GOTO next2;

!obsluha chyby z timeoutu cteni ReadStrBin
ERROR
IF ERRNO =  ERR_DEV_MAXTIME THEN
!vynecha upozorneni (program by spadl a na Pendantu se objevilo varovani
o chybe)
SkipWarn;
!vyskakuje z rutiny do serialRead, kde se kontroluje prichozi zprava
RETURN;
ENDIF
ENDPROC
```



Obr. 22 Vývojový diagram čtení zpráv proměnné délky

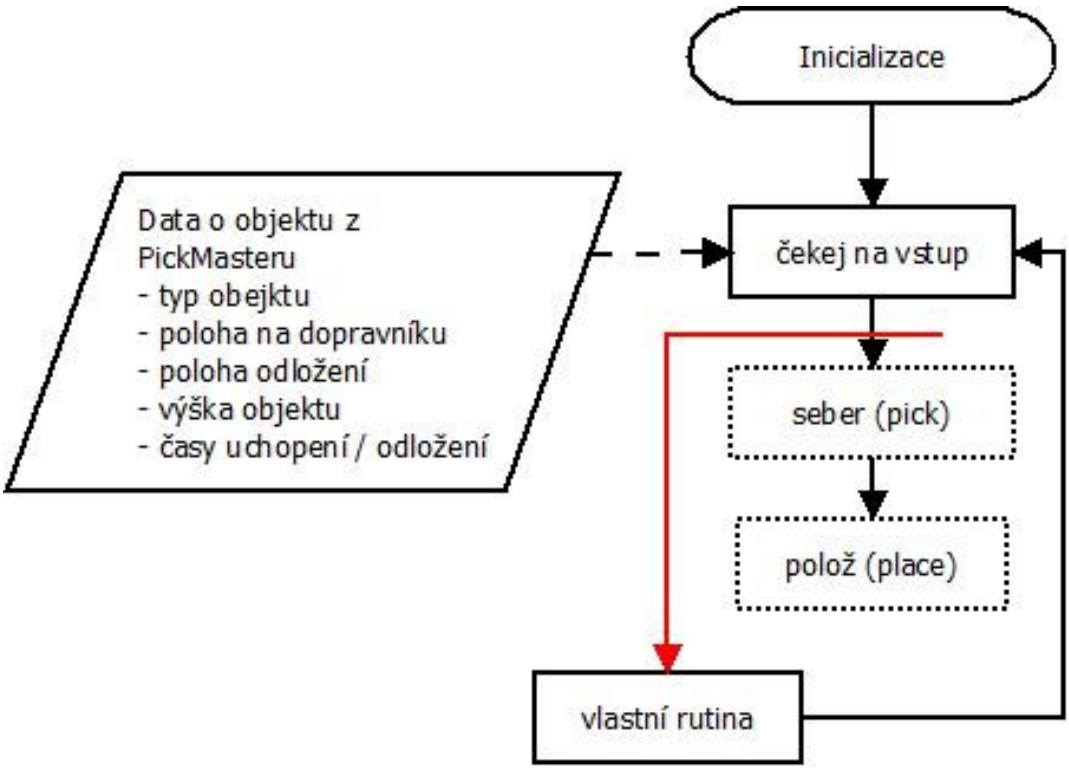
5 Standardní program a vstup do něj



Obr. 23 Vývojový diagram - standardní program

Na Obr. 23 je znázorněn zjednodušený vývojový diagram standardního programu FlexPickeru.

1) Při **inicializaci** programu (stisknutí tlačítka „spustit“ v prostředí PickMaster 3) je do řadiče robotu nahrán RAPID kód, který přísluší danému projektu a je editovatelný přímo z PickMasteru. Součástí tohoto programu jsou i proměnné, jako je rychlost robotu (*TCP speed*) a jiné.
2) Po inicializaci se spustí dopravník a rozpoznávání obrazu. Program nyní **čeká na vstup** z PickMasteru. Ten je vygenerován po rozpoznání objektu (*item*) a obsahuje údaje o poloze objektu na dopravníku, požadované poloze odložení, výšce objektu (aby do něj robot nenarazil), definici časů uchopení a odložení (jak dlouho má robot setrvat na pozici pro bezpečné uchycení / odložení předmětu) a další.
3) Program poté vstupuje do pohybových rutin sebrání (**Pick**) a odložení (**Place**), které ve svých instrukcích používají výše zmíněné informace.
4) Po dokončení pohybu a odložení objektu se postup opakuje pro další rozpoznaný objekt.



Obr. 24 Vývojový diagram - vstup do programu

Chce-li uživatel využít sofistikovaný systém rozpoznávání obrazu a sledování dopravníku systému PickMaster 3 pouze k vyhledávání objektů, ale dále s touto informací nakládat po svém, je nutné toto provést přímo v RAPID kódu. Do programu je nutno vstoupit před standardní rutinou *Pick*, kdy již jsou uloženy všechny potřebné údaje o objektu, viz . Obr. 24. Následují vlastní rutiny, ve kterých jsou tyto údaje využity.

5.1 Popis jednotlivých rutin

A) PPAMAIN

Inicializace programu a sekvence provádění (*execution seq*)

1) **main()**
   Startovní rutina - program vždy začíná na této rutině
2) **InitSafeStop**()
   Inicializace přerušení *SafeStopInt* do servisní rutiny *SafeStopTrap*
3) **InitTriggs()**
   Nastavuje *trigger events* pro ovládání vakua pro každé *item source*
   `IF (ItmSrcData{i}.Used) THEN  SetTriggs i;`
4) **InitPickTune**()
   Inicializace přerušení *PickTuneInt* do servisní rutiny *PickTuneTrap*
5) **SetTriggs**(**num Index**)
   Inicializace *triggerů* pro vakuum podle **`ItmSrcData{Index}.SourceType`**
   *(*Rutinu je nutno editovat, pokud je použito více přísavek.*)*
6) **InitSpeed**()
   Inicializace rychlostních limitů.
7) **PickPlace()**
   Inicializace nastavení na začátku programu a volání rutiny *PickPlaceSeq*.
   Needitovat!
8) **SafeStop()**
   Rutina volaná ze servisní rutiny *SafeStopTrap*
9) **GotoRestartPos()**
   Přesouvá robot do pozice *SafePos* a potvrdí všechny *item sources*, aby mohl být proces restartován.
   Předtím uloží současnou pozici robotu do robtargetu *pCurrentPosition* a jointtargetu *jointpos1***.**
10) **SafeStopTrap**
    Servisní rutina přerušení *SafeStopInt*. Volá rutinu *SafeStop()*, ovládá bezpečné zastavení robotu.
11) **PickTuneTrap**
    Nastaví laděná data (*Vacuum delays, FollowTime, OffZ, ...*)

B) PPASERVICE

Obsahuje servisní rutiny, které se pak dají volat přímo z PickMasteru.

C) PPAEXECUTING

Tento modul řídí všechny pohyby robotu a je možno jej editovat za účelem uživatelského přizpůsobení programu, proto je rutinám zde obsaženým věnována větší pozornost.

1) **SetupIndexes()**

   Slouží k rozřazení *ItmSrcData* do tří skupin podle typu (*Pick, Place, Other*). Tyto se dále používají při volání rutiny *PickPlaceSeq*.

```
! Inicializace promennych
VAR num PickNumber:=1;
VAR num PlaceNumber:=1;
VAR num OtherNumber:=1;

! Prochazi vsemi ItmSrcData{i} a podle SourceType je
! rozrazuje do jednotlivych skupin (Pick, Place, Other)
FOR i FROM 1 TO MaxNoSources DO
IF (ItmSrcData{i}.Used) THEN
IF (ItmSrcData{i}.SourceType = PICK_TYPE) THEN
PickIndex{PickNumber}:=i;
Incr PickNumber;
ELSEIF (ItmSrcData{i}.SourceType = PLACE_TYPE) THEN
PlaceIndex{PlaceNumber}:=i;
Incr PlaceNumber;
ELSE
OtherIndex{OtherNumber}:=i;
Incr OtherNumber;
ENDIF
ENDIF
ENDFOR
```

Příklad:

V programu je pole proměnných o šesti prvcích[3] *ItmSrcData{}* naplněno takto:

- Dva prvky s parametrem SourceType=PICK_TYPE,
- Tři prvky s parametrem SourceType=PLACE_TYPE,
- Jeden prvek s neznámým parametrem SourceType=NEZNAMY_TYPE

[3] Ve skutečnosti má pole *ItmSrcData{}* počet prvků roven proměnné *MaxNoSources*. Uvažujeme, že však pouze 6 prvků pole má atribut **`Used=True`**. Ostatní prvky algoritmus přeskakuje.

ItmSrcData{1}.Sourcetype= PICK_TYPE

ItmSrcData{2}.Sourcetype= PICK_TYPE

ItmSrcData{3}.Sourcetype= PLACE_TYPE

ItmSrcData{4}.Sourcetype= PLACE_TYPE

ItmSrcData{5}.Sourcetype= PLACE_TYPE

ItmSrcData{6}.Sourcetype= NEZNAMY_TYPE

Po proběhnutí rutiny jsou pole naplněna takto:

PickIndex{1}=1

PickIndex{2}=2

PlaceIndex{1}=3

PlaceIndex{2}=4

PlaceIndex{3}=5

OtherIndex{1}=6

2) **PickPlaceSeq()**

Sekvence *Pick-Place*. Tuto rutinu je možno editovat k přizpůsobení pohybů robotu. Jakmile se program PickMasteru dostane až do této rutiny, má již vše inicializováno a nastaveno. Proto je ve všech úlohách v této práci právě z této rutiny volána hlavní rutina daného modifikovaného programu (například *kominkyMain*).

Příklad:

Pokračování s poli *PickIndex{}* a *PlaceIndex{}* naplněnými podle příkladu výše:

```
PROC PickPlaceSeq()
! Provede Pick z ItmSrcData{1}
Pick PickIndex{1};
! Provede Place do ItmSrcData{3}
Place PlaceIndex{1};

! Provede Pick z ItmSrcData{2}
Pick PickIndex{2};
! Provede Place do ItmSrcData{4}
 Place PlaceIndex{2};
ENDPROC
```

3) **Pick(num Index)**

Rutina pro vyzvednutí OM z dopravníku, či zásobníku. Jako parametr (*num Index*) se ji předává číslo příslušného prvku pole *ItmSrcData* (předáno z volající rutiny *PickPlaceSeq*, viz výše).

```
PROC Pick(num Index)

! Nastaveni promennych ridicich MaxTime flag u rutiny GetItmTgt
CONST num nWaitTime:=0.5;
VAR bool bTimeOut;
! Tato rutina se vola pokazde, kdyz se ma robot koordinovat s
dopravnikem
Coordinated;

! Nacte prislusny workobject pro vyzvednuti OM(napr dopravnik)
! Nemusi na nej cekat, protoze je pevne definovan na zacatku
projektu
WObjPick:=ItmSrcData{Index}.Wobj;

! Ceka na udaj o poloze prijizdejiciho OM
GetItmTgt ItmSrcData{Index}.ItemSource,
PickTarget \MaxTime:=nWaitTime \TimeFlag:=bTimeOut;

! Po uplynuti daneho casu se vyskakuje z cekani na udalost,
! aby robot nezustaval zbytecne viset nad odkladacim mistem
IF bTimeOut THEN
! Najede do SafePos (cekaci poloha)- do offsetu
MoveL Offs(SafePos,150,-80,-80),MaxSpeed,fine,Gripper;
! Dal ceka na udaj o poloze prijizdejiciho OM
GetItmTgt ItmSrcData{Index}.ItemSource,PickTarget;
ENDIF

! Najizdi nad OM do vzdalenosti .OffsZ, zapina vakuum
TriggL\Conc,RelTool
(PickTarget.RobTgt,0,0,-ItmSrcData{Index}.OffsZ),
MaxSpeed,ItmSrcData{Index}.VacuumAct1,z20,Gripper\WObj:=WObjPick;

! Najizdi pro OM, zustava na nem po definovany cas .TrackPoint
MoveL\Conc,PickTarget.RobTgt,LowSpeed,
z5\Inpos:=ItmSrcData{Index}.TrackPoint,Gripper\WObj:=WObjPick;

! Nastavuje zatizeni od OM
GripLoad ItemLoad;

! Najizdi nad OM do vzdalenosti .OffsZ, posila potvrzeni .Ack
TriggL RelTool
(PickTarget.RobTgt,0,0,-ItmSrcData{Index}.OffsZ),
LowSpeed,ItmSrcData{Index}.Ack,z20,Gripper\WObj:=WObjPick;

! Vypina koordinaci s dopravnikem
UnCoordinated;
ENDPROC
```

4) **Place(num Index)**

Rutina pro odložení OM do odkládacího zásobníku, nebo na odkládací dopravník. Jako parametr (*num Index*) se ji předává číslo příslušného prvku pole *ItmSrcData* (předáno z volající rutiny *PickPlaceSeq*, viz výše).

Za pozornost zde stojí zejména pohyb pomocí funkce *TriggL*, ve kterém je možno řídit digitální výstupy (zde ovládání vakua a foukání) v definovaném čase ještě před dosažením cílového bodu. Více o funkci *TriggL* v kapitole 5.3.

```
PROC Place(num Index)

! Tato rutina se vola pokazde, kdyz se ma robot koordinovat s
dopravnikem
! Nemusi se pouzivat, pokud je odkladaci misto pevne
Coordinated;

! Nacte prislusny workobject pro odlozeni OM (napr dopravnik nebo
pevny zasobnik)
! Nemusi na nej cekat, protoze je pevne definovan na zacatku
projektu
WObjPlace:=ItmSrcData{Index}.Wobj;

! Nacita udaj o odkladaci pozici, uklada ji jako PlaceTarget
GetItmTgt ItmSrcData{Index}.ItemSource,PlaceTarget;

! Najizdi nad odkladaci misto do vzdalenosti .OffsZ
MoveL\Conc,RelTool(PlaceTarget.RobTgt,0,0,-
ItmSrcData{Index}.OffsZ),MaxSpeed,z20,Gripper\WObj:=WObjPlace;

! Najizdi na odkladaci misto, zustava na nem po definovany cas
.TrackPoint
! Podle casu nastaveneho v triggdata VacuumRev1 (foukani) a
VacuumOff1 (vypnuti vakua/foukani)
! se toto provede jeste pred dosazenim odkladaciho bodu
TriggL\Conc,PlaceTarget.RobTgt,LowSpeed,
ItmSrcData{Index}.VacuumRev1\T2:=ItmSrcData{Index}.VacuumOff1,
z5\Inpos:=ItmSrcData{Index}.TrackPoint,Gripper\WObj:=WObjPlace;

! Vypina nastaveni zatizeni od OM
GripLoad load0;

! Najizdi nad odkladaci misto do vzdalenosti .OffsZ, posila
potvrzeni .Ack
TriggL RelTool(PlaceTarget.RobTgt,0,0,-ItmSrcData{Index}.OffsZ),
LowSpeed,ItmSrcData{Index}.Ack,z20,Gripper\WObj:=WObjPlace;

! Vypina koordinaci s dopravnikem
! Nemusi se pouzivat, pokud je odkladaci misto pevne
UnCoordinated;
ENDPROC
ENDMODULE
```

5.2 Item Source Data

Specifickým datovým typem, se kterým se při práci s PickMasterem uživatel setká, je pole *ItmSrcData*. Toto pole obsahuje nastavení pro všechny pracovní prostory *(Work Areas)*, se kterými se pracuje.

- Jednotlivé pracovní prostory jsou prvky pole
  (např. *ItmSrcData{1}, ItmSrcData{2}...).*
- Jednotlivá nastavení jsou uložena jako sada parametrů
  (např. *ItmSrcData{1}.Trackpoint)*

Tedy každý prvek pole má definovánu vlastní sadu parametrů.

Vybrané parametry pole ItmSrcData

Parametr	Datový typ	Popis
.Used	bool	Prvek pole je používán
.ItemSource	itmsrc	Popis *item source*
.SourceType	itmsrctype	Typ *item source* (Pick / Place / Undefined)
.Ack	triggdata	*Triggdata* pro oznámení o zpracování OM
.VacuumAct1	triggdata	*Triggdata* pro aktivaci vakua
.VacuumRev1	triggdata	*Triggdata* pro aktivaci foukání
.VacuumOff1	triggdata	*Triggdata* pro vypnutí vakua / foukání
.Wobj	wobjdata	*Work Object* pro konkrétní pracovní prostor
.TrackPoint	stoppointdata	Délka setrvání na *Pick / Place* pozici
.OffsZ	num	Výška přibližovacího bodu nad *Pick / Place* pozicí

5.3 Funkce TriggL a její využití při časování pohybů

V kódu v kapitolách výše je použita pohybová funkce *TriggL.* Pro pochopení je nutno vědět, co znamená a jak se používá. Nejprve bude popsáno její použití obecně, poté pro ovládání časování pohybů v pracovních prostorech. Obrázky obecného použití jsou převzaty z [19].

5.3.1 Obecně o funkci TriggL

Pohybovou funkci *MoveL* je vhodné nahradit funkcí *TriggL* v případech kdy uživatel potřebuje:

- přesně řídit akci (viz níže) v případě pohybů s nastavenou zónou (*zone*)
- akci aktivovat v přesně definovaný čas před či po dosažení cílového bodu

Než je tuto funkci možno použít, musí být definován tzv. ***trigger*** (v překladu "spoušť", "spoušťový mechanismus"), například *TriggIO,* viz níže.

Akce, které lze takto v definovaném čase před / za cílovým bodem řídit zahrnují:

- výstup digitálních a analogových signálů - použití *triggeru TriggIO*
- přerušení (*interrupt*) - použití *triggeru TriggInt*
- kontrolu vstupního signálu - například kvůli bezpečnosti - použití *triggeru TriggCheckIO*

Stejně lze takto řídit výše zmíněné akce i během pohybu v kloubech (místo *MoveJ* se používá *TriggJ*) nebo po kružnici (místo *MoveC* použijeme *TriggC*).

Pohybové funkce	**Typy *triggerů***
TriggL	TriggIO
TriggJ	TriggEquip
TriggC	TriggInt
	TriggCheckIO

TriggIO

Tento typ *triggeru* se v pohybových funkcích *TriggL, TriggJ* a *TriggC* používá k přesnému ovládání digitálního nebo analogového signálu během pohybu robotu.

Například, je-li požadováno nastavení digitálního signálu *gun* na hodnotu 1 přesně 0,2s před cílovým bodem *p1*, vypadá kód takto:

```
! Inicializace
VAR triggdata gunon;
! Nastaveni triggeru 0.2 s pred bod
TriggIO gunon, 0.2\Time\DOp:=gun, 1;
! Pouziti triggeru v pohybove funkci
TriggL p1, v500, gunon, fine, gun1;
```

Na Obr. 25 je pak zobrazena trajektorie takového pohybu s kótou určující bod, kdy je signál *gun* přepnut na hodnotu 1.

Obr. 25 Použití funkce TriggL pro definici času sepnutí signálu

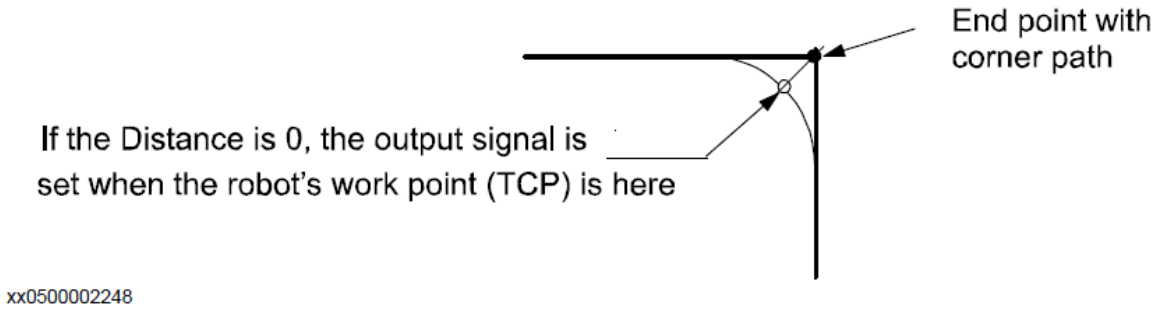
Je-li vhodné namísto časového údaje použít údaj vzdálenosti v mm, vynechá se "*/Time*" a dosadí požadovaná číselná hodnota, například pro 2 mm před cílový bod:

```
! Nastaveni triggeru 5 mm pred bod
TriggIO gunon, 5 \DOp:=gun, 1;
```

Pohybovou funkci s použitím *triggeru TriggIO* (a všech ostatních *triggerů*) je vhodné použít i tehdy, je-li akci nutno provést uprostřed pohybu kolem bodu s nastavenou zónou (*zone*). Pak lze nastavit nulovou, resp. minimální vzdálenost takto:

```
! Nastaveni triggeru do minimalni vzdalenosti
TriggIO gunon, 0 \DOp:=gun, 1;
```

Trajektorie a bod spuštění jsou znázorněny na Obr. 26



Obr. 26 Použití funkce TriggL pro sepnutí signálu při použití zóny

Poznámka:

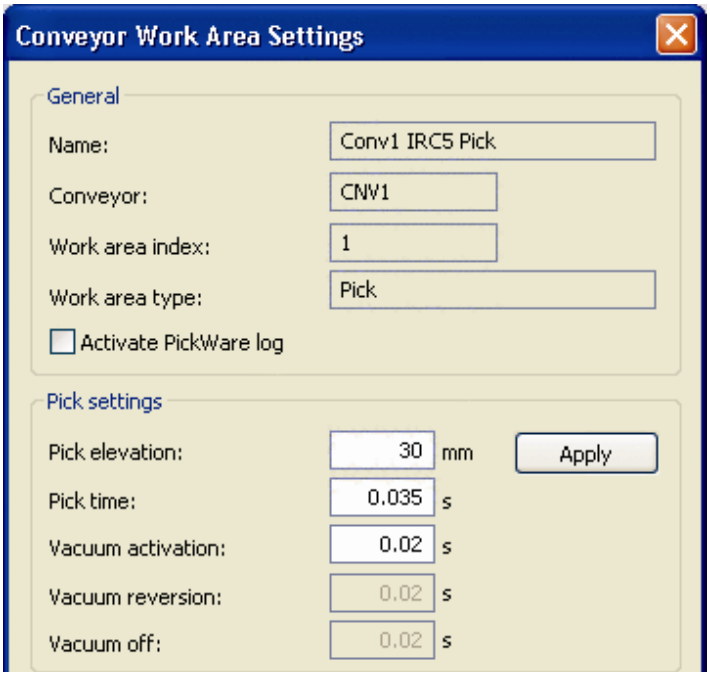
Program PickMasteru používá jiné typy *triggerů* - *TriggEquip*. Ty se při stratu projektu definují v rutině *SetTriggs* v modulu PPAMAIN. Funkce i syntaxe je podobná výše popsanému *TriggIO.*

V příloze [Příloha D] se nachází simulace v RobotStudiu, kde jsou předvedeny definice *triggerů TriggEquip*. Přiložena je rovněž textová verze RAPID kódu této simulace.

5.3.2 Časování při pohybech robotu

Velmi důležité pro optimalizaci procesu je nastavení časování. Toto se provádí přímo v prostředí projektu PickMasteru (dialog nastavení pracovního prostoru, viz Obr. 27), přičemž je možno jej nastavovat i v průběhu procesu.

Na Obr. 28 a Obr. 29 jsou zobrazeny jednotlivé nastavitelné časové parametry. Nad obrázky je popsána vazba těchto parametrů na RAPID kód. V úlohách v kapitolách 6.1 a 6.2 je předvedeno, jak obejít tyto parametry a nastavit je manuálně.

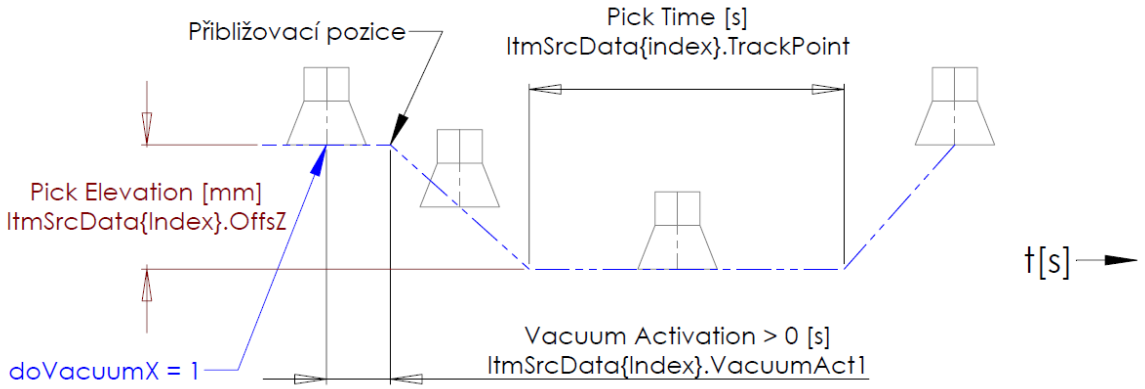


Obr. 27 Dialog nastavení pracovního prostoru

Při **sebrání z dopravníku** (*Pick)* je ještě před dosažením přibližovacího bodu zapnuto vakuum. Vzduchotechnice je tak dán náskok pro vytvoření dostatečného podtlaku ještě před najetím na pozici OM.

Trigger pro zapnutí vakua, tedy čas předstihu a příslušný digitální výstup (viz výše) je definován v *ItmSrcData{Index}.VacuumAct1*. Po najetí do pozice čeká po čas nastavený v *ItmSrcData{Index}.TrackPoint.* Průběh sebrání je schématicky zobrazen na Obr. 28.

```
TriggL\Conc,
RelTool(PickTarget.RobTgt,0,0,-ItmSrcData{Index}.OffsZ),
MaxSpeed,ItmSrcData{Index}.VacuumAct1,z20,Gripper\WObj:=WObjPick;

MoveL\Conc,PickTarget.RobTgt,LowSpeed,
z5\Inpos:=ItmSrcData{Index}.TrackPoint,Gripper\WObj:=WObjPick;
```

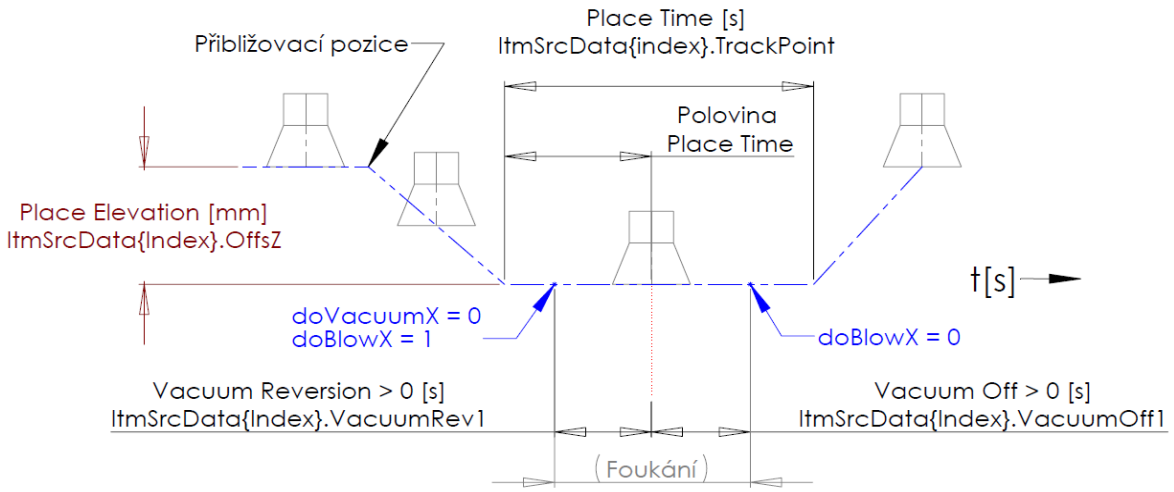


Obr. 28 Průběh a časování při sebrání (*Pick*)

Při **odložení** (*Place)* najíždí robot přes přibližovací pozici do odkládací pozice, na které opět zůstává po čas definovaný v *ItmSrcData{Index}.TrackPoint.* Zde je důležitý okamžik uplynutí poloviny tohoto času (dále *poločas Pick*).

Před *poločasem Pick* je vypnuto vakuum a spuštěno foukání. To umožňuje rychlejší pokles podtlaku v systému a tím rychlejší uvolnění OM. Po *poločase Pick* je vypnuto i foukání. Průběh odložení je schématicky zobrazen na Obr. 29.

```
TriggL\Conc,PlaceTarget.RobTgt,LowSpeed,
ItmSrcData{Index}.VacuumRev1\T2:=ItmSrcData{Index}.VacuumOff1,
z5\Inpos:=ItmSrcData{Index}.TrackPoint,Gripper\WObj:=WObjPlace;
```



Obr. 29 Průběh a časování při odložení (*Place*)

5.4 Způsoby editace RAPID kódu

Editace RAPID kódu v systému FlexPicker se liší od editace u samostatného robotu. Existuje několik možných přístupů. Znalost postupů, jejich úskalí, výhod a nevýhod je zásadní pro další práci.

Nejprve je nutno popsat rozdíl mezi typy modulů a to, jak s nimi PickMaster zachází.

Programové moduly (*Program modules*)

Při použití s PickMasterem jsou po spuštění projektu přemazány kódem modulů projektu, které jsou přístupné přes textový editor. Mažou se všechny původně načtené programové moduly, nejen ty se shodným názvem.

Ukládají a načítají se samostatně (přípona *.mod*), jako program (přípona *.pgf* nebo *.prg)*, nebo jako součást zálohy (*Backup*).

Systémové moduly (*System modules*)

Zůstávají stále v řadiči i po nahrání nového programu (přípona *.pgf* nebo *.prg)*, ani PickMaster je nemaže. Toho je využito v postupu 5.4.3.

Ukládají a načítají se samostatně (přípona *.sys*), nebo jako součást zálohy (*Backup*). Neukládají se s programem.

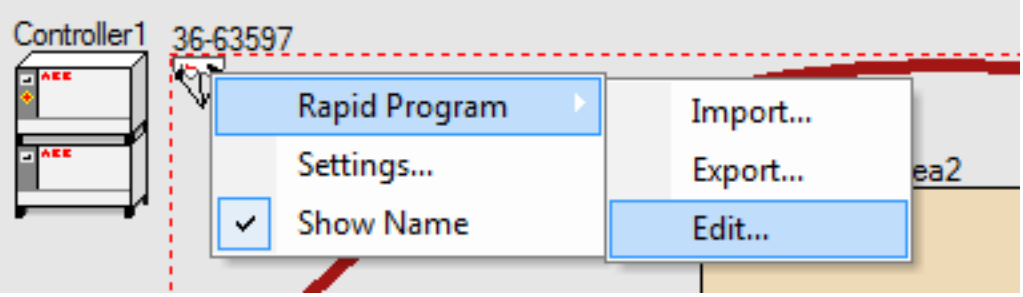
5.4.1 Editace přímo v prostředí PickMaster 3 a textovém editoru

Tvůrci softwaru pravděpodobně preferovaná varianta, avšak nejméně uživatelsky příjemná. Editace se provádí v běžném textovém editoru (Word, WordPad, Poznámkový blok, Notepad++), který samozřejmě nemá možnost kontroly syntaxe. Notepad++ však umí díky filtrům dostupným na internetu alespoň obarvit text (klíčová slova, aritmetické operace, rutiny atd.) Špatná syntaxe se projeví až chybovou hláškou při spuštění projektu a chyba se hledá často velmi složitě.

Tento postup je vhodný jen pro menší úpravy RAPID kódu, případně pro zkušené uživatele, kteří nedělají chyby v syntaxi.

1) V prostředí PickMaster 3 kliknout pravým tlačítkem na symbol robotu, dále Rapid Program - Edit (viz Obr. 30)
2) RAPID kód se otevře v textovém editoru (výchozí editor je možno zvolit v nastavení programu). Tento textový soubor je pouze dočasný (*Temporary - Temp*).

3) Po jeho úpravě je nutno jej v textovém editoru uložit a po návratu do PickMasteru potvrdit dialogové okno, zda jej opravdu použít. Tím se text uloží do projektu PickMasteru a otevřený dočasný textový soubor s RAPID kódem pozbude platnost. Pro další editaci je nutno jej znovu otevřít podle bodu 1).



Obr. 30 Vstup do editace RAPID kódu

Zjednodušeně:
Otevřít textový editor s RAPID kódem, editovat, uložit, potvrdit v PickMasteru.

Výhody:

+ snadný přístup k RAPID kódu
+ není potřeba spouštět RobotStudio

Nevýhody:

– bez možnosti kontroly syntaxe
– nepřehlednost RAPID kódu

5.4.2 Editace v dočasných modulech v RobotStudiu

Tato možnost je pracnější než přímá editace v textovém editoru, avšak nabízí výhodu možnosti kontroly syntaxe. Je vhodná i pro rozsáhlejší úpravy RAPID kódu, nebo pro uživatele, kteří si nejsou jistí syntaxí.

1) Podle postupu výše otevřít textový editor s RAPID kódem.
2) V RobotStudiu se připojit k řadiči robotu (karta *Online)*
3) Ve stromě v levém panelu rozbalit RAPID moduly (*název robotu - RAPID - T_ROB1 - Program modules*)

4) Pokud je nahrán program z předchozí instance PickMasteru, ponechat programové moduly PPAMAIN, PPASERVICE, PPAEXECUTING. Jestliže tam nejsou, vytvořit je (PTM na *T_ROB1* - *New module...*). Pokud je nahrán jiný program, nejdříve jej uložit, aby se předešlo ztrátě dat (PTM na *T_ROB1* - *Save Program As*...).
5) Systémové moduly (*System modules*) s předponou "SYS_" jeden po druhém uložit, aby se předešloztrátě dat (PTM na konkrétní modul - *Save Module As...*) a následně taktéž smazat. Jsou pozůstatkem použití metody popsané níže.
6) Zkopírovat moduly z textového editoru do příslušných modulů v RobotStudiu.
7) Nyní je možno při editaci v RobotStudiu kontrolovat syntaxi (tlačítko *Apply*).
8) Po editaci a kontrole kódu jej zkopírovat zpět do textového editoru. Programové moduly v RobotStudiu se po spuštění projektu smažou!
9) Textový soubor uložit a potvrdit v PickMasteru.

Zjednodušeně:

Otevřít textový editor s RAPID kódem, zkopírovat do RobotStudia, editovat, zkopírovat zpět, uložit, potvrdit v PickMasteru.

Výhody proti postupu výše:

+ možnost kontroly syntaxe
+ přehlednost RAPID kódu

Nevýhody:

− možnost opomenutí zkopírovat zpět před spuštěním projektu - ztráta editovaného programu
− vyšší pracnost

5.4.3 Použití systémových modulů

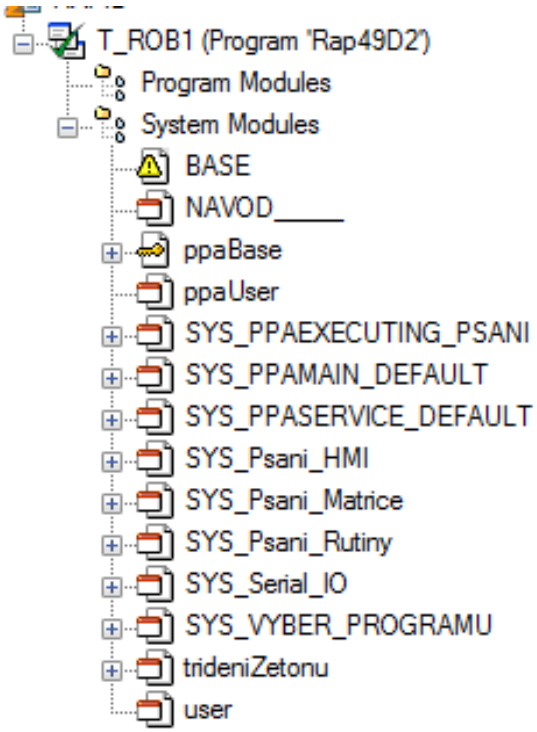
Tento postup využívá toho, že PickMaster při spuštění projektu systémové moduly nemaže. Je vhodný i pro velmi rozsáhlé změny v RAPID kódu, případně jeho rozšíření dalšími moduly (viz všechny úlohy v kapitole 6). Naprosto při tom obchází textovou verzi RAPID kódu v projektu PickMasteru, což však na druhé straně vede k nutnosti udržovat v modulech, jejich názvech a organizaci předem daný systém. Po odladění úlohy je možno vše přesunout zpět do projektu PickMasteru (kap. 5.4.1).

Pokud tato metoda nebyla použita dříve, postup je následující:

1) Otevřít v PickMasteru textový editor s RAPID kódem původního, nebo málo upraveného projektu.
2) V RobotStudiu vytvořit systémové moduly (*T_ROB1 - New Module... - Module Type: System)* s předponami "SYS_" a vhodně zvolenými příponami pro snadnější rozlišitelnost, tedy například:

   SYS_PPAMAIN_DEFAULT (pokud se nebude lišit od původního)
   SYS_ PPASERVICE_DEFAULT (pokud se nebude lišit od původního)
   SYS_ PPAEXECUTING_ULOHA1 (pokud bude editován)
   SYS_ULOHA1 (nový samostatný modul, pokud je potřeba)

Na Obr. 31 je zobrazen strom pro úlohu skládání znaků.



Obr. 31 Příklad vzhledu stromu systémových modulů

3) Do těchto modulů zkopírovat příslušné části RAPID kódu z textového editoru.

4) Systémové moduly si uložit (PTM na modul - *Save Module As...).*

5) V textovém editoru vše smazat, uložit a potvrdit v PickMasteru.

6) Moduly editovat přímo v RobotStudiu, při spuštění projektu se do programových modulů nic nenahrává (RAPID kód u projektu je prázdný). Moduly si průběžně ukládat.

7) Je vhodné vytvořit jeden systémový modul, ve kterém budou jen poznámky o modulech potřebných pro všechny úlohy. Při každé nové úloze je nutné příslušné moduly načíst a nepotřebné smazat.

8) Po odladění programu je možno moduly nakopírovat zpět do textového souboru příslušného projektu PickMasteru. Obsahy všech modulů se jednoduše nakopírují za sebe včetně deklarací modulů (MODULE nazevmodulu ... ENDMODULE). Při kopírování je nutno smazat parametr *(SYSMODULE)* za deklarací modulu.

Zjednodušeně:

Otevřít textový editor s RAPID kódem, zkopírovat kód do systémových modulů v RobotStudiu, smazat RAPID kód v textovém editoru, uložit, potvrdit v PickMasteru, editovat systémové moduly. Po odladění případně zkopírovat zpět.

Výhody:

+ možnost kontroly syntaxe
+ přehlednost RAPID kódu
+ je možné projekt rychle ladit - projekt se okamžitě spouští z PickMasteru bez nutnosti kopírování kódu do textového editoru
+ jednoduché rozšíření o další moduly

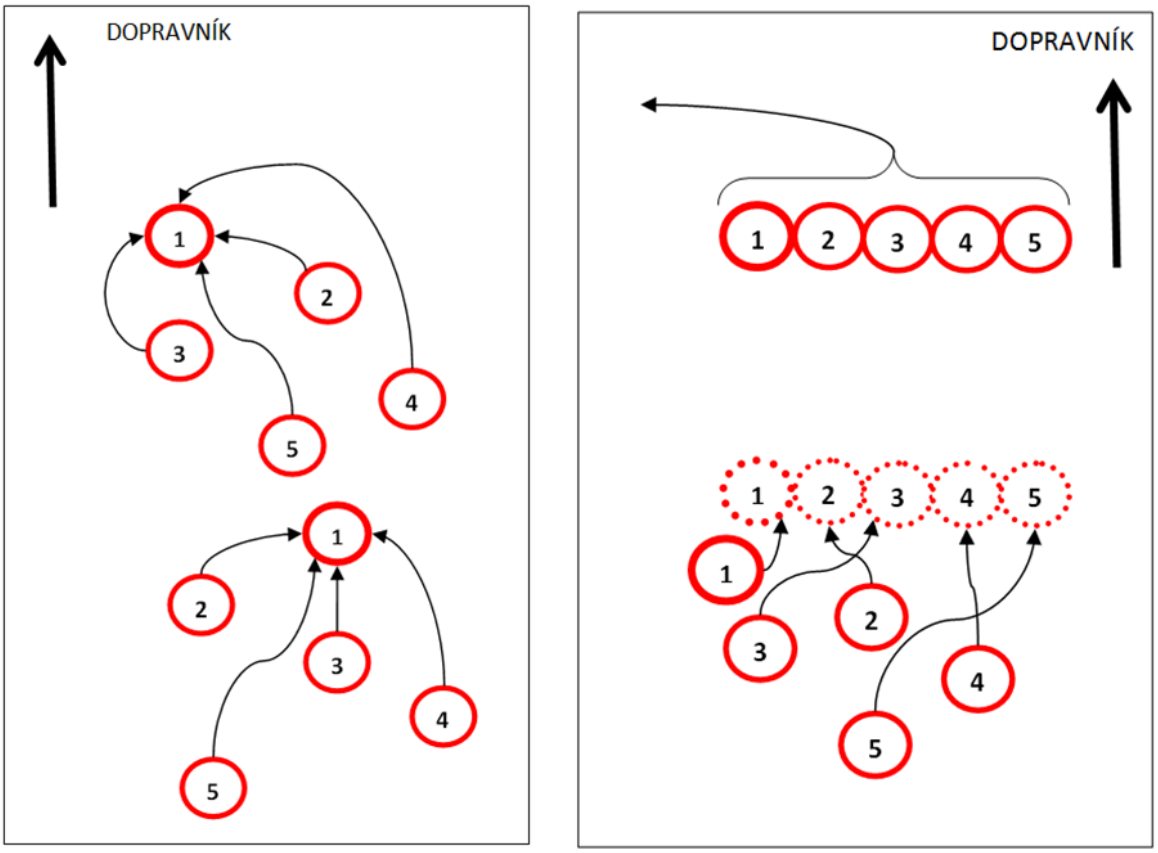
Nevýhody:

− nutnost udržování pořádku v modulech
− pro změnu programu v robotu je nutné změnit moduly

6 Úlohy k realizaci

6.1 Skládání objektů do komínků

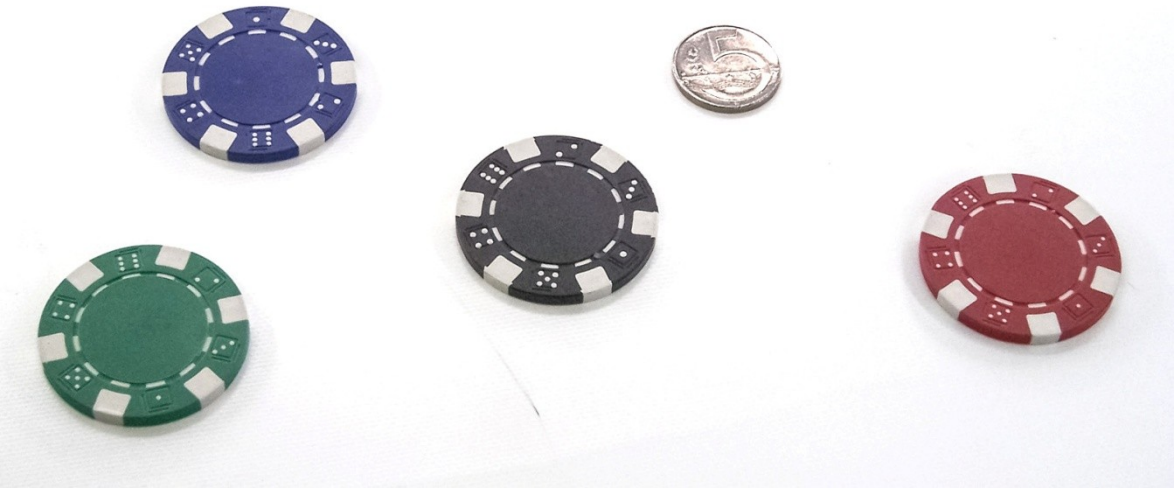
Poměrně jednouchou úlohou, na které lze postup vstupu do programu předvést, je naskládání objektů do „komínků“ na pozici prvního příchozího objektu. Názorně je tato úloha zobrazena na Obr. 32 vlevo. Je předvedeno i manuální ovládání vakua - vlastní definice *triggerů* (viz kapitola 5.3).



Obr. 32 Skládání do komínků a vedle sebe

Takový postup lze v praxi s výhodou použít tam, kde je nutno zkrátit manipulační časy a jsou k dispozici alespoň dva roboty v sérii za sebou. První robot vhodně uspořádá objekty na dopravníku (na sebe či vedle sebe, viz Obr. 32), druhý robot pak sebere celou skupinu objektů a přesune ji. Výhoda je tedy v tom, že první robot operuje jen v malém prostoru, tedy na krátkých drahách. Druhý robot pak sice na drahách delších, ale s menším počtem cyklů. Tím lze významně ušetřit manipulační čas. Nevýhoda je ta, že druhý robot musí mít efektor přizpůsobený pro sbírání většího počtu objektů (vícero přísavek, mechanický efektor apod.). Jelikož na LCR pracujeme pouze s jedním robotem, tyto „komínky“ pokračují dále po dopravníku, na jehož konci padají do připravené krabice.

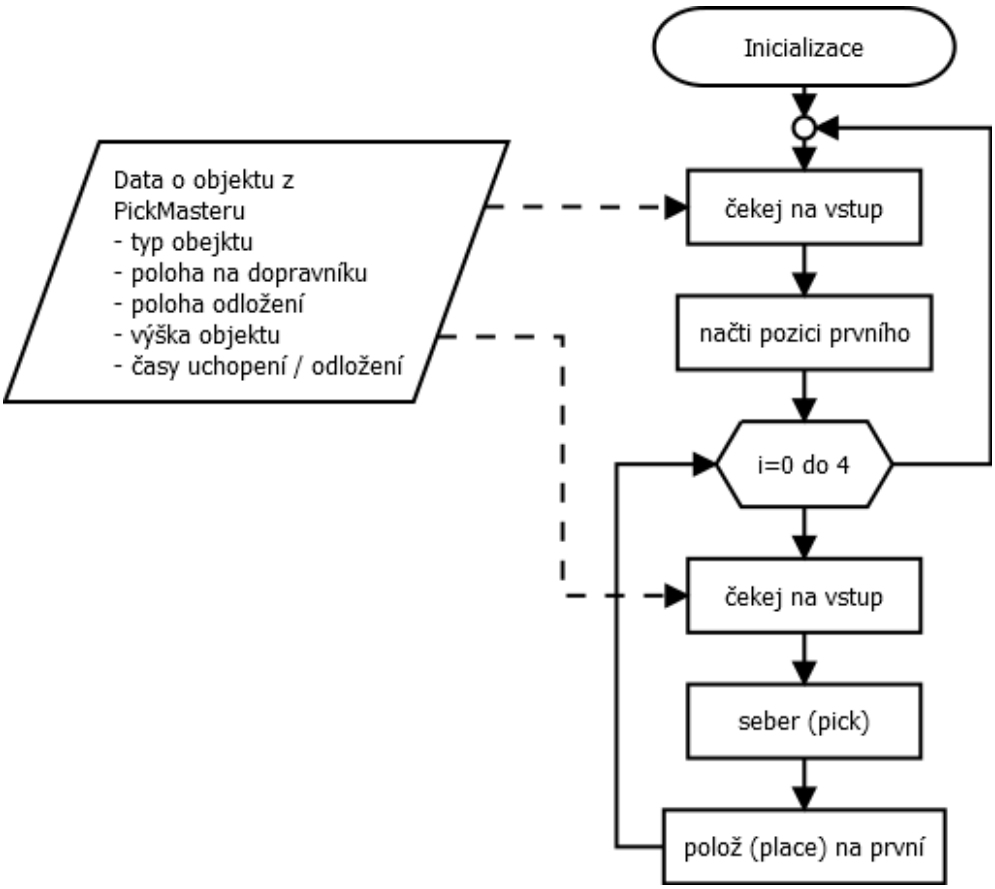
Pro potřeby realizace této úlohy byly zakoupeny barevné pokerové žetony, viz Obr. 33. Žetony mají čtyři barevné varianty - modrou, zelenou, červenou a černou a je tedy možné je podle těchto barev třídit, jak je ukázáno v další úloze (kap. 6.2). V této úloze barva rozpoznávána není.



Obr. 33 Žetony

Na Obr. 34 je znázorněn zjednodušený vývojový diagram této úlohy pro umístění skupiny pěti objektů na pozici prvního (pro jiný počet objektů se změní počet opakování cyklu FOR).

1) Po rozpoznání prvního objektu (zárodku pole) je tento ponechán na dopravníku na místě, je pouze načtena jeho pozice.
2) Další objekty skupiny (zde další 4 objekty) jsou po rozpoznání sebrány (*Pick*) a uloženy (*Place*) relativně k pozici prvního objektu. Může jít o skládání na sebe (poloha objektů bude mít odlišnou souřadnici Z), nebo vedle sebe (odlišné souřadnice X a Y, avšak stále pevně vztažené k poloze prvního objektu), viz Obr. 32.
3) Po uspořádání požadovaného množství objektů přechází program opět na začátek, kde čeká na údaje o pozici dalšího „zárodku“ pole.



Obr. 34 Vývojový diagram skládání do komínků

Program obsahuje 4 samostatné moduly. V době psaní této práce je program funkční ve formě systémových modulů s názvy s předponou "SYS_" (viz kapitola 5.4.3). Dále se mohou tyto moduly objevit i bez této předpony.

1) **SYS_PPAMAIN_DEFAULT -** základní modul PickMasteru
2) **SYS_PPASERVICE_DEFAULT -** základní modul PickMasteru
3) **SYS_PPAEXECUTING_KOMINKY** - obsahuje změnu, kdy v rutině *PickPlaceSeq* ihned volá hlavní rutinu programu psaní *kominkyMain*
4) **SYS_KOMINKY** - hlavní rutiny pro úlohu skládání do komínků - inicializace (*initKominky*), hlavní rutina (*kominkyMain)* a manipulace s OM (*kominkyPick, kominkyPlace*).

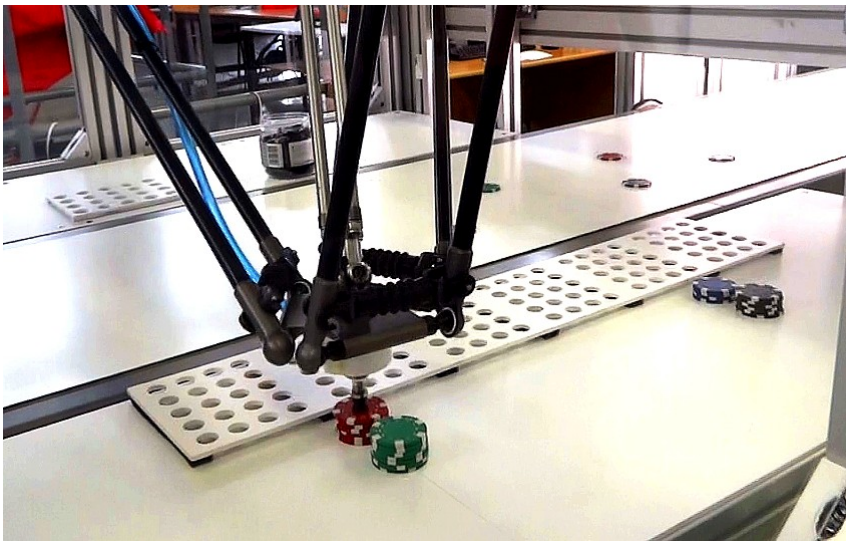
V příloze [Příloha F] je zahrnut program ve formě zálohy (*Backup*), jednotlivé moduly v textové podobě a projekt PickMasteru. V příloze [Příloha I] pak demonstrační video.

6.2 Třídění žetonů podle barev

Další úlohou je třídění objektů podle barvy.

- Barva OM je rozpoznána pomocí kamerového systému.
- Robot třídí OM podle barvy na pevná odkládací místa (odkládací stolečky po stranách dopravníku).
- Roztříděné OM jsou spočítány a výsledek je zobrazován na FlexPendantu.

Úloha je opět demonstrací vstupu do programu a jeho vlastního dalšího řízení. Program je řízen vlastní rutinou hned za vyzvednutím OM z dopravníku. Je předvedeno i manuální ovládání vakua - vlastní definice *triggerů* (viz kapitola 5.3)



Obr. 35 Úloha třídění žetonů podle barvy

Program obsahuje 4 samostatné moduly. V době psaní této práce je program funkční ve formě systémových modulů s názvy s předponou "SYS_" (viz kapitola 5.4.3). Dále se mohou tyto moduly objevit i bez této předpony.

1) **SYS_PPAMAIN_DEFAULT -** základní modul PickMasteru
2) **SYS_PPASERVICE_DEFAULT -** základní modul PickMasteru
3) **SYS_PPAEXECUTING_TRIDENI_ZETONU** - obsahuje změnu, kdy v rutině *PickPlaceSeq* ihned volá hlavní rutinu programu psaní *trideniMain*
4) **SYS_TRIDENI_ZETONU** - hlavní rutiny pro úlohu třídění žetonů - inicializace (*initTrideni*), hlavní rutina (*trideniMain)* a manipulace s OM (*trideniPlace*)

V příloze [Příloha G] je zahrnut program ve formě zálohy (*Backup*), jednotlivé moduly v textové podobě a projekt PickMasteru. V příloze [Příloha I] pak demonstrační video.

6.3 Skládání znaků do matrice

Jako finální úloha bylo navrženo skládání nápisů do matrice. Na této úloze jsou předvedeny pokročilé možnosti práce s cykly, poli, sériovou komunikací a uživatelským vstupem přes rozhraní FlexPendantu.

- Uživatel může zvolit nápis (maximálně 6 znaků) - vstup pomocí rozhraní na FlexPendantu nebo přes mobilní telefon a Bluetooth modul.
- Robot tento nápis vyskládá z vhodných objektů manipulace (dále OM) do připravené matrice (pro stabilizaci odložených OM).
- OM se sbírají z dopravníku, kde jsou rozpoznány kamerovým systémem.
- Nepotřebné OM robot odkládá do připraveného zásobníku nebo na skluz.
- Pro zvýšení efektivity robot využívá vyrovnávací zásobník (*Buffer*).
- Při přeskládávání písmene je vyhledáván nejbližší volný OM v matrici (zvýšení efektivity).
- Proces je možno provádět i manuálně přes FlexPendant pomocí předdefinovaných příkazů (např. pro zvednutí z písmene 1, řádku 2, sloupce 3 - příkaz "pck123"), viz v kapitole 6.3.5.
- V programu je veden záznam o tom, které pozice jsou (i manuálně) obsazené.

Program pro tuto úlohu je velmi komplexní (cca 600 řádků RAPID kódu bez mezer), v následujících kapitolách jsou proto popsány jen jeho základní principy a uzly. K jeho dalšímu studiu lze využít celý jeho kód v příloze [Příloha H], nebo lépe, simulaci v RobotStudiu v příloze [Příloha E]. V příloze [Příloha I] je pak demonstrační video.

Jako vhodné objekty manipulace byly vybrány a zakoupeny kameny pro hru GO (Obr. 36).

Obr. 36 Kameny GO

Pro stabilizaci OM byla navržena a vyrobena matrice, viz Obr. 37.

Obr. 37 Matrice pro skládání znaků

Projekt v PickMasteru obsahuje pouze základní prvky, tedy nad rámec linky jen *Position Source 1* (pro dopravník), *Position Source 2* (obecné odkládací místo) a jediný objekt manipulace (*item*) - kámen GO definovány metodou naučení (*Train*).

6.3.1 Moduly úlohy skládání znaků do matrice

Program obsahuje 7 samostatných modulů. V době psaní této práce je program funkční ve formě systémových modulů s názvy s předponou "SYS_" (viz kapitola 5.4.3). Dále se mohou tyto moduly objevit i bez této předpony.

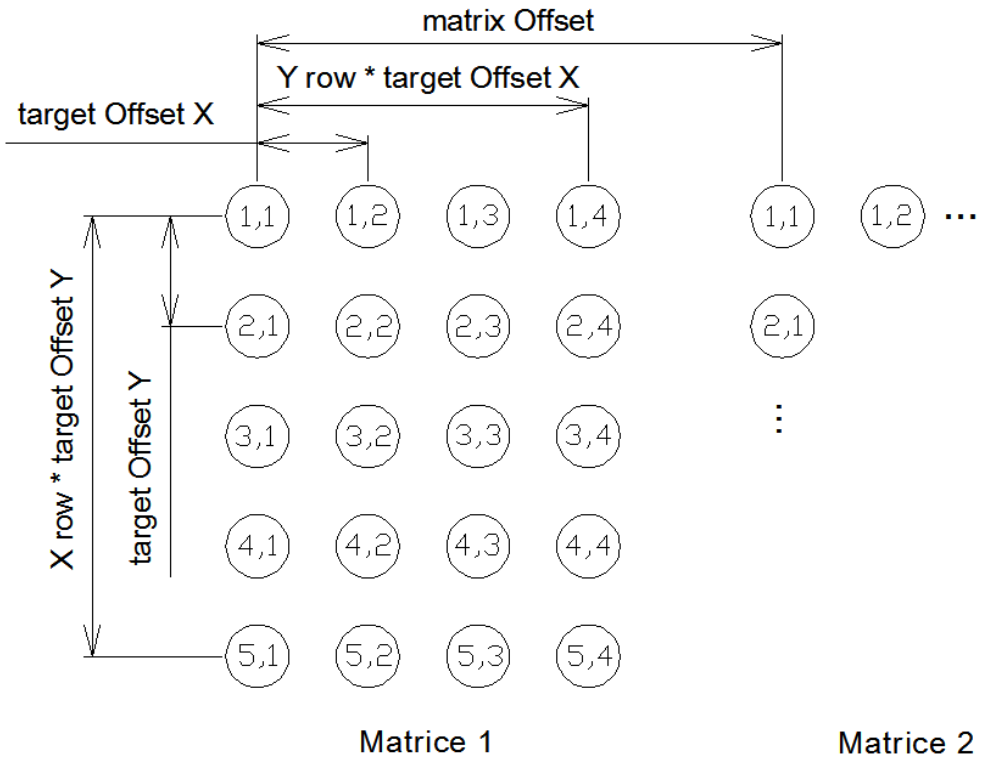
1) **SYS_PPAMAIN_DEFAULT -** základní modul PickMasteru
2) **SYS_PPASERVICE_DEFAULT -** základní modul PickMasteru
3) **SYS_PPAEXECUTING_PSANI** - obsahuje změnu, kdy v rutině *PickPlaceSeq* ihned volá hlavní rutinu programu psaní *psaniMain*
4) **SYS_Psani_Rutiny** - všechny hlavní rutiny pro úlohu psaní od inicializace, přes manipulaci s OM (*pick, place*) až po samotné algoritmy psaní znaků
5) **SYS_Psani_Matrice** - obsahuje rutinu výběru znaku a definice vzhledu všech znaků
6) **SYS_Psani_HMI** - uživatelské rozhraní na FlexPendantu
7) **SYS_Serial_IO** - sériová komunikace s Bluetooth modulem a její vyhodnocení
8) **SYS_showPath** - předváděcí pohybová rutina

6.3.2 Matrice a princip práce s nimi

Protože by bylo velmi nepříjemné pro každou pozici v matricích definovat samostatný *robtarget* a následně do něj robot polohovat, jsou využita pole. RAPID umí do jedno- i vícerozměrných polí ukládat všechny své datové typy, včetně *robtargetů.* Toho je v této úloze využito jak pro prvotní definici pozic *robtargetů* v prostoru, tak pro jejich následné adresování. Nutno poznamenat, že bez využití polí *robtargetů* by tato úloha byla jen stěží realizovatelná.

V hlavičce modulu *"SYS_Psani_Rutiny"* jsou definovány základní proměnné určující vzhled matric. Na Obr. 38 je znázorněn význam jednotlivých proměnných.

```
CONST num wordLength:=5;              !pocet znaku (matric)
CONST num Xrow:=5;                    !pocet radku matrice
CONST num Yrow:=4;                    !pocet sloupcu matrice
CONST num bufferLength:=5;            !pocet pozic v bufferu
CONST num targetOffsetX:=25;          !roztec sloupcu
CONST num targetOffsetY:=25;          !roztec radku
CONST num matrixOffset:=115;          !roztec matric (pro pDispPose)
CONST num bufferOffset:=35;           !roztec pozic v bufferu
CONST num pickOffset:=40;             !zdvih nad pozici
```



Obr. 38 Význam proměnných rozměrů matric

Jediné, co je tak nutno definovat, je první prvek v matrici a v *bufferu*. Protože jsou pohyby robotu vztaženy ke konkrétním *workobjectům (Workobject_Matrix* a *Workobject_Buffer*, viz níže), je první prvek pole definován vždy v nulových souřadnicích příslušného *workobjectu* (viz vyplňování polí níže). Souřadnice prvního prvku (*robtargetu*) jsou tak shodné se souřadnicemi příslušného *workobjectu*. V hlavičce se tedy manuálně zadává poloha *workobjectu,* vůči kterému jsou jeho *robtargety* definovány.

```
PERS wobjdata Workobject_Matrix:=
[FALSE,TRUE,"",[[-332,-294, -1140],[1,0,0,0]],[[0,0,0],[1,0,0,0]]];
PERS wobjdata Workobject_Buffer:=
[FALSE,TRUE,"",[[-211, 325, -1149.5],[1,0,0,0]],[[0,0,0],[1,0,0,0]]];
```

Pole *robtargetů* příslušných rozměrů:

```
VAR robtarget tArrayMatrix {Xrow,Yrow};          !pole RT na matrici
VAR robtarget tArrayBuffer {bufferLength};       !pole RT bufferu
```

se pak v cyklech *FOR* vyplňují takto:

```
PROC initTargets()
!naplni 2D pole RT na matrici
for i from 1 to Xrow do
for j from 1 to Yrow do
tArrayMatrix{i,j}:=[[(j-1)*targetOffsetY,-(i-1)*targetOffsetX,0],
[0,1,0,0],[0,0,0,0],[0,0,0,0,0,0]];
endfor
endfor
!naplni 1D pole RT na bufferu
for i from 1 to bufferLength do
tArrayBuffer{i}:=[[(i-1)*bufferOffset,0,0],
[0,1,0,0],[0,0,0,0],[0,0,0,0,0,0]];
endfor
ENDPROC
```

Výsledkem je pole *robtargetů* pro **jednu** matrici. Při přechodu do další matrice se používá stejné pole, ale celý program se posouvá o rozteč matric rutinou:

```
PROC setMatrix(num matrix)
! nastavi prislusny posun
currPDispPose:=pDispPoseArray{matrix};

!aktivuje posun programu
PDispSet currPDispPose;
ENDPROC
```

kde *pDispPoseArray* je opět pole, tentokrát vyplněné v rutině *initDispPoseArray* podle zadaných roztečí matric *matrixOffset*.

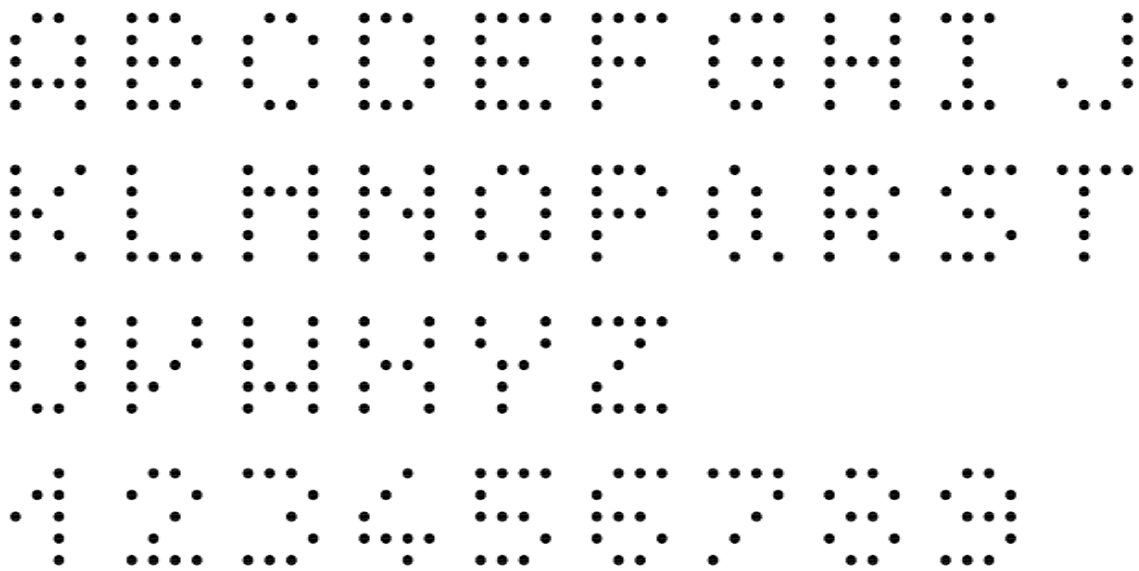
Práce s takto vytvořenými poli je pak již jednoduchá. Například pro vyzvednutí (*pick*) objektu manipulace z konkrétní matrice (například 1.) a místa (2. řádek, 3. sloupec) vypadá postup zjednodušeně takto:

1) Nastavit pořadí matrice
   ```
   setMatrix 1;
   ```
2) Volat rutinu pro zvednutí - rychlost přesunu MaxSpeed, nájezd k pozici a odjezd z pozice rychlostí LowSpeed (odvozeny od maximální rychlosti nastavené v projektu PickMasteru)
   ```
   pickMatrix 2,3, MaxSpeed, LowSpeed;
   ```

V rutině *pickMatrix* se pak do pohybu (např. *MoveL*) jako *robtarget* přiřadí konkrétní prvek pole *robtargetů tArrayMatrix{2,3}* .

6.3.3 Znaky a jejich definice

Bylo navrženo písmo pro všechny kapitální znaky a číslice (Obr. 39). Z důvodu úspory místa byla zvolena matrice 4x5 (řádky x sloupce).



Obr. 39 Vzhled znaků

Znaky jsou definovány ve formě číselných polí *symbolMatrix*. Při požadavku na zapsání znaku je toto pole nejprve načteno (rutina *chooseSymbol*) a následně jsou OM vloženy na všechny pozice, které mají hodnotu 1 a odstraněny z pozic, které mají hodnotu 0, viz v další kapitole.

```
VAR num symbolMatrix{5,4};

PROC chooseSymbol(string symbol)
TEST symbol
CASE "A":
      symbolMatrix:=
[     [0,1,1,0],
      [1,0,0,1],
      [1,0,0,1],
      [1,1,1,1],
      [1,0,0,1]];
...
ENDTEST
ENDPROC
```

6.3.4 Algoritmus skládání znaku

Samotný algoritmus skládání znaku je obsažen v rutině *writeSymbol*. Její zjednodušený vývojový diagram je na Obr. 40. Nejprve se rutinou *chooseSymbol* zvolí příslušný vzhled znaku a všechny znaky se odemknou (viz níže). Poté se ve FOR cyklech postupně procházejí všechny pozice na matrici a vyhodnocuje se, zda tam OM být má, či nemá.

Pro vyhodnocování slouží další dvě pole:

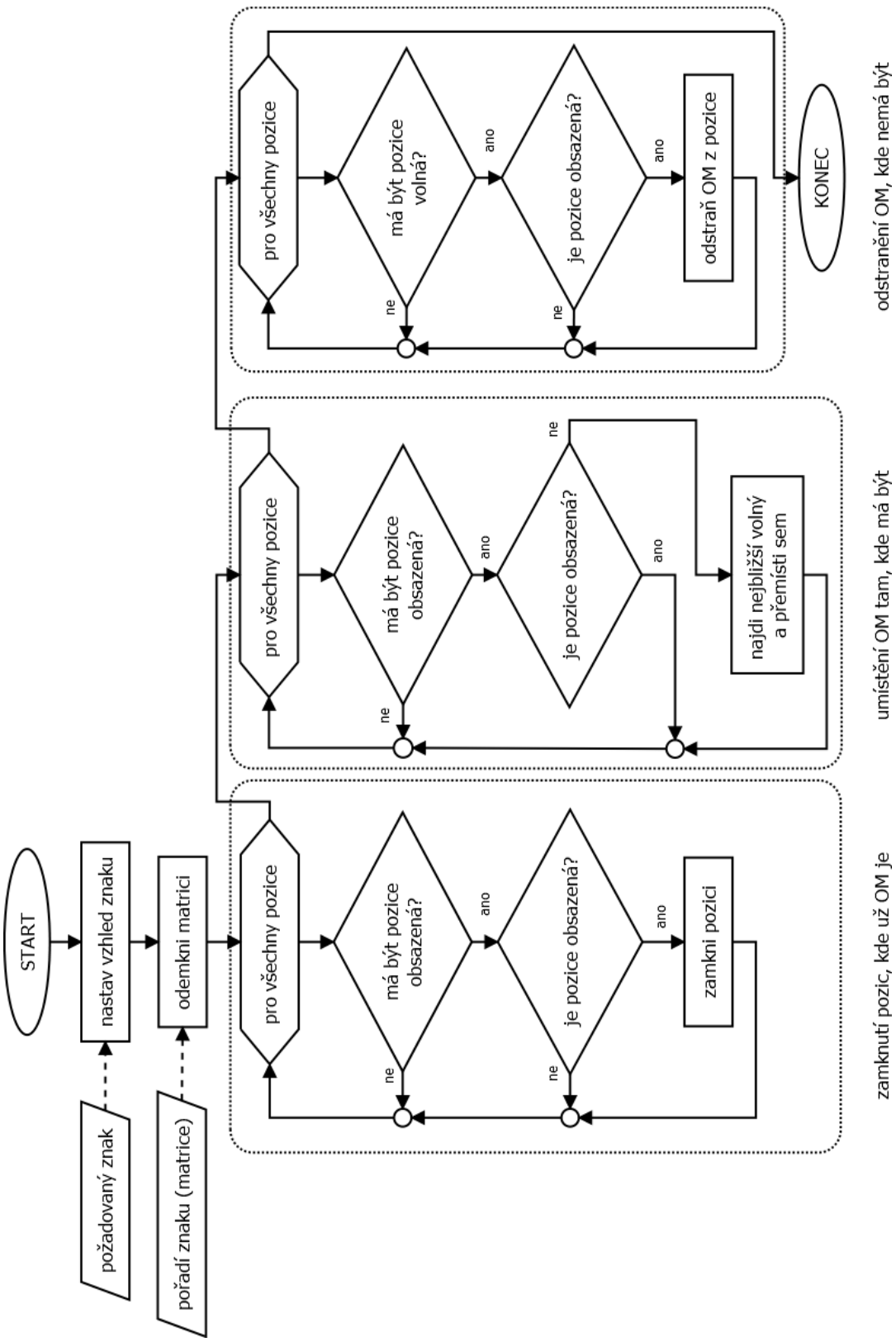
- `VAR bool bArrayPresent {wordLength,Xrow,Yrow}`
  - 3D pole - obsahuje informaci, zda se na daném místě nachází OM
  - pro všechny pozice ve všech matricích {pořadí matrice, č. řádku, č. sloupce}
  - aktualizuje se pokaždé, když se naplní, či vyprázdní pozice

- `VAR bool bArrayLocked {wordLength,Xrow,Yrow}`
  - 3D pole - obsahuje informaci, že se na dané pozici má nacházet OM a že se tam již nachází
  - pro všechny pozice ve všech matricích {pořadí matrice, č. řádku, č. sloupce}
  - na začátku skládání znaku dojde k odemknutí všech pozic v matrici rutinou *unlockMatrix*
  - na to se zamykají všechny pozice, kde OM má být (podle nastaveného vzhledu znaku) a už je (porovnáním s polem *bArrayPresent*)
  - dále se aktualizuje po každém vložení OM do matrice při skládání znaku

Zamykání pozic je nezbytné pro správnou funkci rutiny *findClosestAndPlace.* Pokud na pozici má být OM a není, je volána právě tato rutina, která prochází v rámci dané matrice všechny pozice, na kterých se nachází OM (*bArrayPresent=true*) a není zamčený, tzn. nemá tam být (*bArrayLocked=false*). Rutina ale nesebere první nalezený OM, nýbrž projde všechny a pomocí Pythagorovy věty vyhodnotí, který je nejblíže. To vede k dalšímu zvýšení efektivity procesu.

Pokud není nalezen žádný volný OM v matrici, je sebrán ze zdroje. Tím je přednostně vyrovnávací zásobník (*buffer*). Pokud se v tomto nenachází žádný OM, je sebrán z dopravníku. Zde je jediný bod, kdy program využívá funkcionality PickMasteru.

Po vyskládání OM tam, kde mají být, se odebírají z pozic, kde být nemají. Přednostně jsou opět ukládány do *bufferu*. Přítomnost OM na pozici v *bufferu* je opět sledována pomocí pole *bArrayBuffer.* Pokud je *buffer* plný, nadbytečný OM se vyhazuje a již jej dále není možno v programu použít.

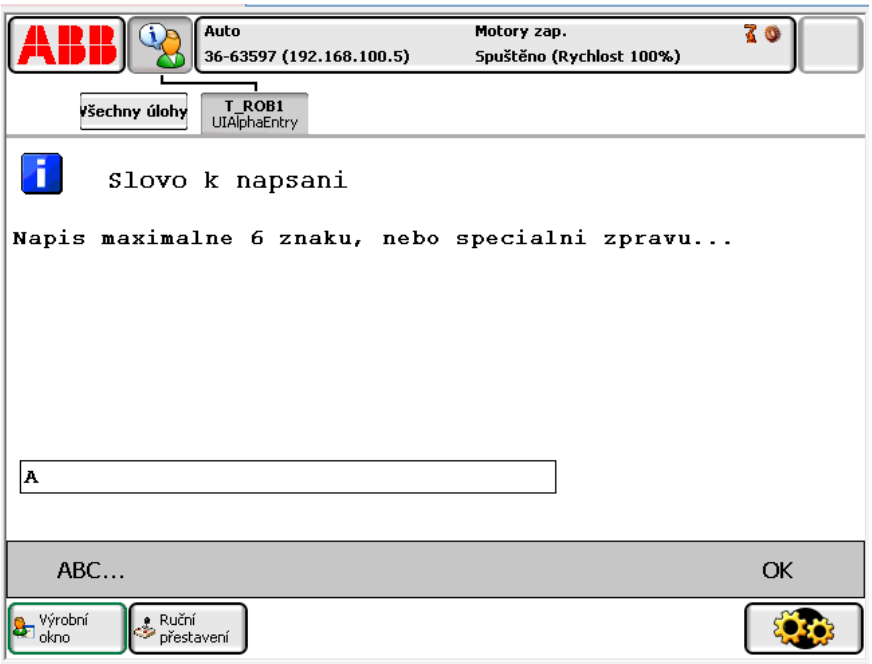


Obr. 40 Vývojový diagram skládání znaku

6.3.5 HMI na FlexPendantu

Pokud je v nastavení programu zadána možnost vstupu z FlexPendantu, po spuštění projektu se na něm zobrazí uživatelský textový vstup (Obr. 41).

```
!vstup textu.. 1= FlexPendant, 2= BT, 3= Path_10
CONST num textInput:=1;
```



Obr. 41 Textový vstup na FlexPendantu

Zde je pak připravena řada příkazů:

- text **VELKÝMI PÍSMENY** - vyskládá nápis do matrice
- "**sho**" - předváděcí rutina (*showPath* v modulu *SYS_showPath*)
- "**abb**" - postupně vyskládá nápisy "ABB","IRB360","FLEX","PICKER"
- "**kat**" - postupně vyskládá nápisy "KAT354","VSBTUO","J MZIK"
- "**cup**" - úklid (*clean up*) - vyhodí všechny OM z matrice, bufferu i dopravníku
- "**src**" - sebere OM z dopravníku (*source*)
- "**tsh**" - vyhodí OM (*trash*)
- "**pck** + č. matrice + řádek + sloupec" (např. "pck321") - sebere OM z matrice
- "**plc** + č. matrice + řádek + sloupec" (např. "plc321") - vloží OM do matrice
- "**buf** + č. pozice" (např. "buf3") - vloží OM do *bufferu*

Tyto příkazy jsou definovány v rutině *fpTextInput* v modulu *SYS_Psani_HMI.*

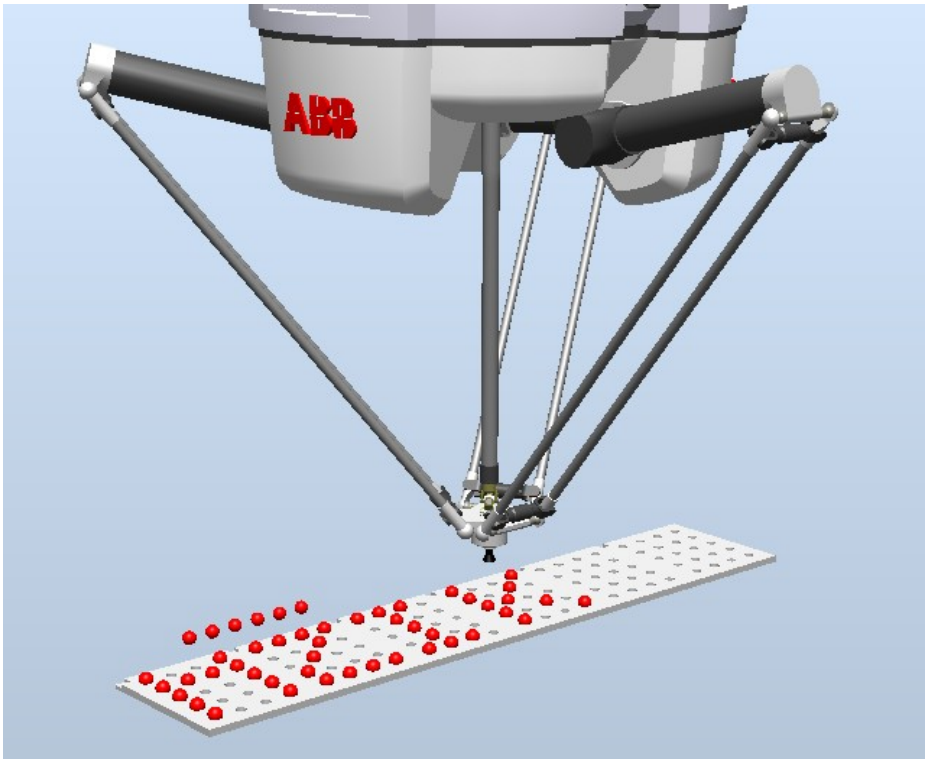
6.3.6 Simulace v RobotStudiu

Při vývoji úlohy skládání znaků byla nejprve vytvořena simulace v RobotStudiu (Obr. 42). Ta obsahuje:

- Simulaci ovládání přes virtuální FlexPendant
- Většinu programového obsahu reálné aplikace, včetně využití *bufferu*
- *Smart Component* přísavky
- *Smart Component* zdroje OM - kuliček

Na této simulaci byly vytvořeny a odladěny všechny principy úlohy skládání znaků, lze ji tedy využít pro jejich hlubší studium. Moduly byly posléze převedeny do reálné aplikace.

Je zahrnuta v příloze [Příloha E] včetně videonávodu, jak ji spustit a demonstračního videa.

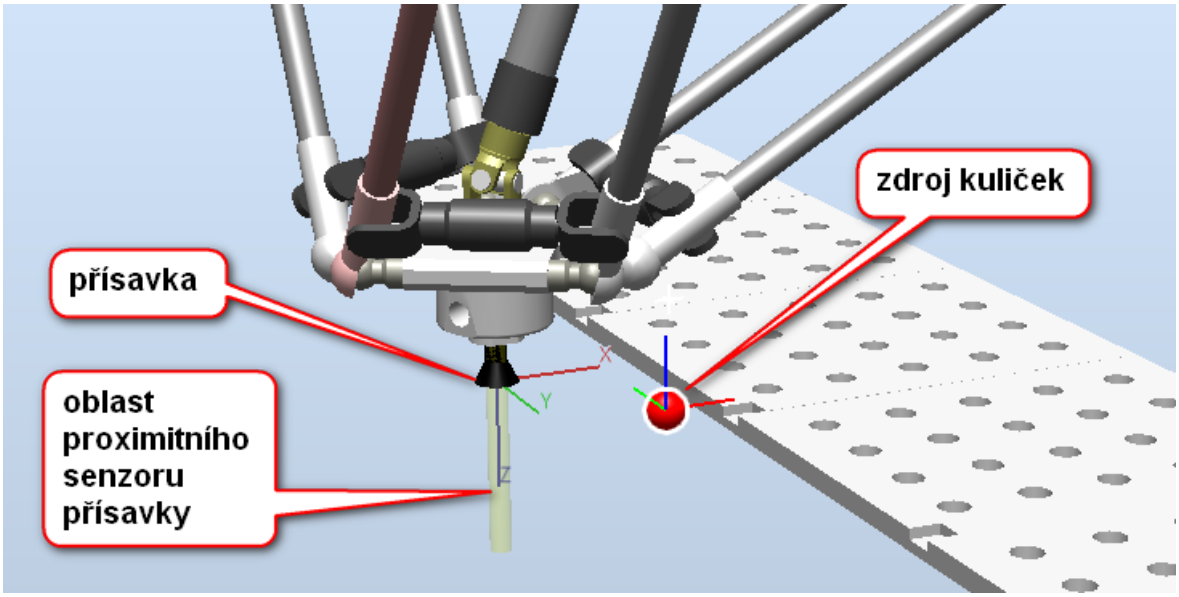


Obr. 42 Simulace v RobotStudiu

Smart Componenty

Stejně jako v reálné aplikaci, i v simulaci v prostředí RobotStudia je proces skládání znaků pružný - závislý na uživatelském vstupu. Protože by bylo neúměrně náročné definovat pro každou kuličku (simulaci OM) události (*Events*) pro její připojení k přísavce (*Attach),* odpojení (*Detach*), zobrazení a mizení (*Visibility*), jsou použity tzv. Smart Components (chytré komponenty). Ty, umožňují tvorbu jejich vnitřní logiky (AND, OR, ...), simulace senzorů (proximitní senzory, hledání nejbližšího komponentu podle zadaného typu...), časovačů, generátor objektů atd.

Nutno však poznamenat, že jejich definice není příliš uživatelsky příjemná a nezkušenému uživateli může cesta k dosažení požadované funkce trvat i několik hodin. Bez nich by však vznik této simulace v takovém rozsahu nebyl možný.



Obr. 43 Smart Componenty v simulaci

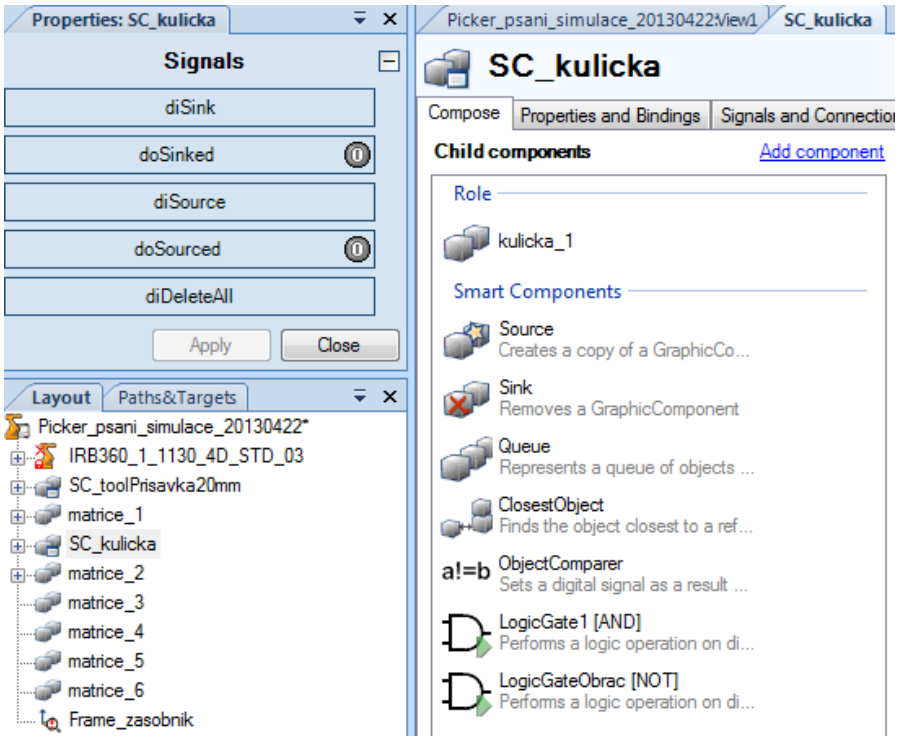
Smart Component - zdroj kuliček

Prvním Smart Componentem je zdroj kuliček. Má následující funkce:

- generace nové kuličky (reakce na signál *diSource*)
- potvrzení, že kulička byla vygenerována (posílá signál *doSourced*)
- vymazání kuličky (reakce na signál *diSink*)
- potvrzení, že kulička byla vymazána (posílá signál *doSinked*)
- vymazání všech kuliček (reakce na signál *diDeleteAll*)

Po aktivaci příslušného signálu ze simulace se tedy na místě zdroje buď objeví nová kulička, nebo se vymaže kulička, která je tam přítomna. Na začátku a konci procesu se vymažou všechny dosud vygenerované kuličky.

Prvky tohoto Smart Componentu a jeho řídicí signály jsou na Obr. 44.



Obr. 44 Smart Component - zdroj kuliček

Smart Component - přísavka

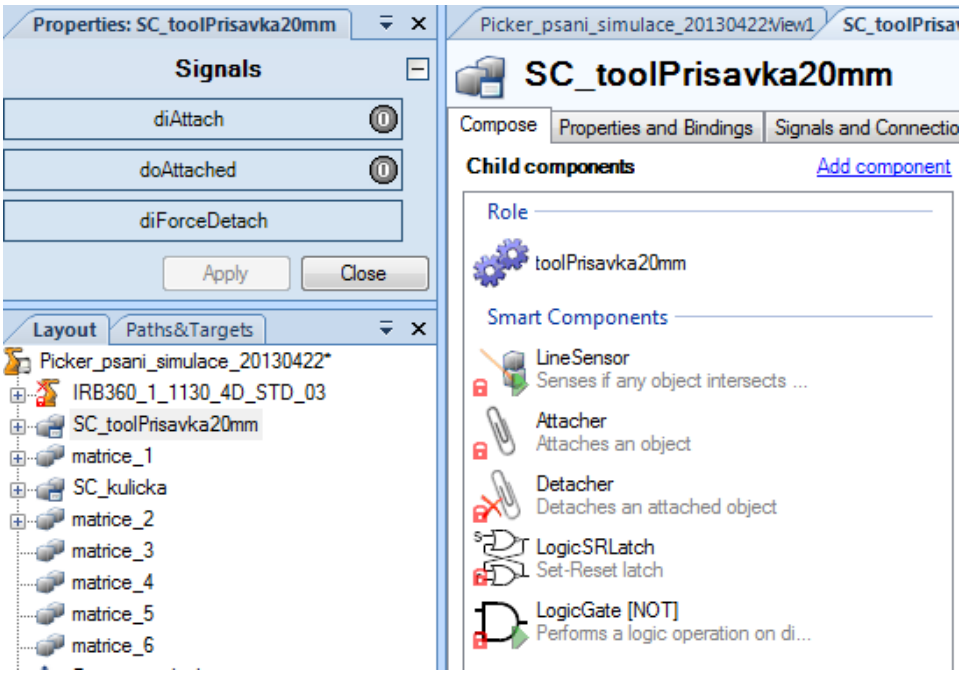
Druhým Smart Componentem je efektor - přísavka. Má následující funkce:

- připojení OM (reakce na signál *diAttach*)
- potvrzení, že byl připojen nějaký OM (posílá signál *doAttached*)
- vynucené odpojení OM (reakce na signál *diForceDetach*)

Po aktivaci příslušného signálu ze simulace se připojí objekt, který se nachází pod efektorem v oblasti proximitního senzoru (*Line Sensor*).

Pro správnou funkci je nezbytné, aby se v dosahu senzoru nacházel pouze jeden objekt (zde kulička). Proto je pro každé sebrání ze zdroje vygenerována nová kulička a po odložení do zdroje je kulička vymazána.

Prvky tohoto Smart Componentu a jeho řídicí signály jsou na Obr. 45.



Obr. 45 Smart Component - přísavka

7 Závěr

Diplomová práce se zabývá pokročilými metodami programování robotu ABB IRB 360 FlexPicker v součinnosti se softwarem ABB PickMaster 3. V úvodu jsou popsány základní prvky robotizovaného pracoviště na Centru robotiky Katedry robototechniky VŠB-TU Ostrava (kapitola 3). Dále je popsána konstrukce a zapojení dodatečně navržených a instalovaných periferií - světelného boxu, odkládacích stolů, Bluetooth modulu a panelu ovládání pracovních prostorů (kapitola 4). Byly vybrány a zakoupeny vhodné objekty manipulace - barevné pokerové žetony a kameny pro hru GO.

Hlavní částí je pak popis průběhu standardního programu a metody vstupu do něj za účelem uživatelských modifikací (kapitola 5). Tyto metody jsou demonstrovány na dvou úlohách napodobujících použití v praxi. Jsou popsány tři postupy editace RAPID kódu (kapitola 5.4). Jako nejlepší se jeví metoda přechodu do systémových modulů, která umožňuje vytvoření i velmi komplexních úloh - zejména úlohy 3 se stovkami vlastních řádků RAPID kódu, viz níže a v kapitole 6.3.

První úloha (kapitola 6.1) - skládání objektů do komínků - je ekvivalentem reálného použití, kde je nutno zkrátit manipulační časy a jsou k dispozici alespoň dva roboty v sérii za sebou. První robot vhodně uspořádá objekty na dopravníku, druhý robot pak sebere celou skupinu objektů a přesune ji. Jelikož na LCR pracujeme pouze s jedním robotem, tyto komínky pokračují dále po dopravníku, na jehož konci padají do připravené krabice.

Druhá úloha (kapitola 6.2) - třídění žetonů podle barvy - demonstruje možnost rozpoznávání barvy a třídění podle ní.

Obě tyto úlohy jsou zejména demonstrací vstupu do programu PickMasteru a jeho dalšího vlastního řízení. Program je řízen vlastními rutinami hned za vyzvednutím OM z dopravníku. Je předvedeno i manuální ovládání vakua - vlastní definice *triggerů.*

Jako třetí úloha (kapitola 6.3) je realizováno vyskládání znaků (psaní) do matrice s možností uživatelského vstupu pomocí FlexPendantu nebo mobilního telefonu a rozhraní Bluetooth. Jde o efektní interaktivní úlohu, která je používána pro prezentace jak tohoto pracoviště, tak Katedry robototechniky a VŠB-TU Ostrava. Vzhledem ke svému rozsahu a složitosti není jako celek určena pro výukové účely. Jsou na ní však demonstrovány funkce, jako práce s cykly, poli, sériová komunikace a uživatelský vstup přes rozhraní FlexPendantu. Tato úloha byla vyvinuta v prostředí RobotStudio, simulace využívá Smart Componenty. Simulace je přiložena.

Všechny úlohy jsou detailně popsány a v elektronické podobě je přiložena jejich kompletní dokumentace. Rovněž jsou přiložena videa těchto úloh.

8 Zdroje

[1] ČSN 01 6910 Úprava písemností psaných strojem nebo zpracovaných textovými editory. Praha : Český normalizační institut, 1997.

[2] ČSN ISO 690 Bibliografické citace. Obsah, forma a struktura. Praha : Český normalizační institut, 1996.

[3] Application manual Pickmaster 3.3. *CourseCast | Panopro Inc.* [Online] [Citace: 2. listopad 2012.] http://lecture.sccsc.edu/amt/AMT%20206%20Electricity%20and%20Automation/ABB%20IRB%20140%20Robot/files/3HAC031978-001_revB_en.pdf.

[4] SICK DFS60B-S4PA10000 - Inkrementální enkodér. *TME.* [Online] [Citace: 22. Duben 2013.] http://www.tme.eu/cz/details/dfs60b-s4pa1000/enkodery/sick/dfs60b-s4pa10000/.

[5] JX Inverter. *OMRON.* [Online] [Citace: 30. duben 2013.] http://www.technidrive.co.uk/media/uploads/JX%20inverter%20brochure.pdf.

[6] Čtecí hlavy CES. *EUCHNER.* [Online] [Citace: 22. duben 2013.] http://www.euchner.cz/produkty/bezpecnost/bezpecnostni-systemy/cteci-hlavy-ces/.

[7] Vyhodnocovací jednotky CES-AZ. *EUCHNER.* [Online] [Citace: 22. duben 2013.] http://www.euchner.cz/produkty/bezpecnost/bezpecnostni-systemy/ces-az/.

[8] ABB IRB 360. *ABB.* [Online] [Citace: 1. listopad 2012.] http://www.abb.com/product/seitp327/cf1b0a0847a71711c12573f40037d5cf.aspx.

[9] IRB 360 FlexPickerTM datasheet. *ABB.* [Online] [Citace: 1. listopad 2012.] http://www05.abb.com/global/scot/scot241.nsf/veritydisplay/e4712d3c88fd9240c125772e005b361b/$file/IRB%20360%20ROB0082EN_E.pdf.

[10] Robots Rule Packaging Processes. *foodprocessing-technology.com.* [Online] [Citace: 9. březen 2013.] http://www.foodprocessing-technology.com/features/feature52679/feature52679-2.html.

[11] PickMaster - Snadné balení. *ABB.* [Online] [Citace: 2. listopad 2012.] http://www.abb.com/product/seitp327/6d52bccc823a4833c1256fdc00282a12.aspx.

[12] **ABB.** *Application Manual DeviceNet.* [DVD dodané k robotu] Document ID: 3HAC020676-001, Revision M.

[13] Bluetooth 2.1+EDR UART, I²C, GPIO Serial Port Adapter moduly. *spezial electronic.* [Online] [Citace: 1. březen 2013.] http://www.spezial.cz/connectblue/bluetooth-2-1-edr-uart-spi-i2c-gpio-modul.html.

[14] 7805 TO220 celoplastové pouzdro. *GM Electronic.* [Online] [Citace: 1. březen 2013.] http://www.gme.cz/linearni-regulatory-napeti-pevne/7805-to220-celoplastove-pouzdro-p330-001/#dokumentace.

[15] CB-OBS 433 Electrical Mechanical Data Sheet. *connectBlue.*
[Online] [Citace: 30. duben 2013.]
http://support.connectblue.com/spaces/flyingpdf/pdfpageexport.action?pageId=6029322.

[16] Bluetooth Serial Port Adapter Toolbox - Getting Started. *connectBlue.*
[Online] [Citace: 1. březen 2013.]
http://www.spezial.cz/pdf/Bluetooth_Serial_Port_Adapter_Toolbox_-_Getting_Started.pdf.

[17] Bluetooth Serial Port Adapter Security. *connectBlue.*
[Online] [Citace: 30. duben 2013.]
http://www.connectblue.com/fileadmin/Connectblue/Web2006/Products/Bluetooth/Gen4/OEM/Common/Documents/Bluetooth_Serial_Port_Adapter_Security.pdf.

[18] Carriage return. *Wikipedia.* [Online] [Citace: 23. březen 2013.]
http://cs.wikipedia.org/wiki/Carriage_return.

[19] **ABB.** *Technical reference manual - RAPID Instructions, Functions and Data Types.*
[DVD dodané k robotu] Document ID: 3HAC 16581-1, Revision J.

[20] **ABB**. *Technical reference manual - RAPID Overview.*
[DVD dodané k robotu] Document ID: 3HAC16580-1, Revision J.

[21] Application manual Pickmaster 3.3. *CourseCast | Panopro Inc.*
[Online] [Citace: 2. listopad 2012.]
http://lecture.sccsc.edu/amt/AMT%20206%20Electricity%20and%20Automation/ABB%20IRB%20140%20Robot/files/3HAC031978-001_revB_en.pdf.

[22] Bluetooth OEM Serial Port Adapter- Product Brief OBS433. *connectBlue.* [Online]
[Citace: 30. duben 2013.] http://www.spezial.cz/pdf/Product_Brief_OBS433.pdf.

[23] **ABB.** *Product Manual IRB 360.* [DVD dodané k robotu]
Document ID: 3HAC030005-001, Revision G.

[24] **ABB**. *PickMaster 3 Application manual.* [DVD dodané k robotu]
Document ID: 3HAC031978-001, Revision F.

[25] **ABB**. *PickMaster 3 Product specification.* [DVD dodané k robotu]
Document ID: 3HAC5842-12, Revision E.

[26] **ABB**. *Product Specification IRB 360.* [DVD dodané k robotu]
Document ID: 3HAC029963-001, Revision K.

9 Seznam příloh

Všechny přílohy jsou pouze v elektronické podobě.

[Příloha A]:

Připojení Bluetooth modulu přes sériový port k PC a jeho konfigurace pro spárování s mobilním telefonem.pdf

[Příloha B]:

Postup kalibrace kamery s robotem ABB 360 FlexPicker a softwarem PickMaster 3.pdf

[Příloha C]:

Picker - psaní - tahák pro exkurze.pdf

[Příloha D]:

Simulace - test triggeru TrigEquip

Obsahuje:

- Pack&Go balíček stanice v RobotStudiu (.rspag)
- RAPID kód simulace (.pdf)

[Příloha E]:

Simulace - psaní znaků do matrice

Obsahuje:

- Pack&Go balíček stanice v RobotStudiu (.rspag)
- videonávod na spuštění simulace psaní znaků (.wmv)

[Příloha F]:

Úloha 1 - skládání do komínků

Obsahuje:

- RAPID kód systémových modulů programu (.pdf)
- zálohu systému (backup) - obsahuje systémové moduly (.sys)
- projekt PickMasteru (.pmproj)
- příslušnou linku v PickMasteru (.pmline)

[Příloha G]:

Úloha 2 - třídění podle barvy

Obsahuje:

- RAPID kód systémových modulů programu (.pdf)
- zálohu systému (backup) - obsahuje systémové moduly (.sys)
- projekt PickMasteru (.pmproj)
- příslušnou linku v PickMasteru (.pmline)

[Příloha H]:

Úloha 3 - psaní znaků do matrice

Obsahuje:

- RAPID kód systémových modulů programu (.pdf)
- zálohu systému (backup) - obsahuje systémové moduly (.sys)
- projekt PickMasteru (.pmproj)
- příslušnou linku v PickMasteru (.pmline)

[Příloha I] :

Videa

Obsahuje:

- Skladani zetonu na kominky.mp4
- Trideni zetonu podle barvy.mp4
- Simulace psani.wmv
- Uloha psani - ABB IRB360 FLEX PICKER.mp4
- Uloha psani - KAT354 VSBTUO J MZIK.mp4
- Vyklizení pracovního prostoru.mp4